Ann. Henri Poincaré 10 (2010), 1335–1358

1424-0637/10/071335-24, *published online* January 13, 2010
DOI 10.1007/s00023-009-0019-6

Annales Henri Poincaré

Hilbert Lattice Equations

Norman D. Megill and Mladen Pavičić

Abstract. There are five known classes of lattice equations that hold in every infinite dimensional Hilbert space underlying quantum systems: generalised orthoarguesian, Mayet's $\mathcal{E}_A$, Godowski, Mayet–Godowski, and Mayet's E equations. We obtain a result which opens a possibility that the first two classes coincide. We devise new algorithms to generate Mayet–Godowski equations that allow us to prove that the fourth class properly includes the third. An open problem related to the last class is answered. Finally, we show some new results on the Godowski lattices characterising the third class of equations.

1. Introduction

In 1995, Solèr [1,2] proved that an infinite-dimensional Hilbert space can be recovered from an orthomodular lattice (OML) together with a small number of additional conditions, with the only ambiguity being that its field may be real, complex, or quaternionic. Specifically, any OML that is complete, is atomic, satisfies a superposition principle, has height at least 4, and has an infinite set of mutually orthogonal atoms, which completely determines such a Hilbert space. This provides us with a dual, purely lattice-theoretical way to work with the Hilbert spaces of quantum mechanics. In addition to offering the potential for new insights, the lattice-theoretical approach may be computationally efficient for certain kinds of problems in quantum mechanics, particularly if, in the future, we are able to exploit what may be a "natural" fit with quantum computation.

However, the approach cannot be applied straightforwardly because unlike the equations (identities) defining OML, the additional conditions needed to recover Hilbert space are first- and second-order quantified conditions. Quantified conditions can complicate computational work: trivially, a computer cannot scan infinite lattices or an infinite number of lattices to determine if "there exists" and/or "for all" conditions are satisfied; more generally, quantified theorem-proving algorithms may be needed to achieve rigorous results. Thus, it is desirable to find equations that can partially express some of these quantified conditions, allowing them to be weakened or possibly even replaced. The goal is to get as close as possible to a purely equational description of $\mathcal{C}(\mathcal{H})$ (the lattice of closed subspaces of a Hilbert space $\mathcal{H}$), in other words to find smaller and smaller equational varieties that contain it.

Until 1975, the only lattice equations known to hold in $\mathcal{C}(\mathcal{H})$ were those defining OML itself. Then Alan Day discovered that a stronger equation, the orthoarguesian law, also holds. There have been several advances since then. In 2000, Megill and Pavičić [3] discovered an infinite family of equations that generalised the orthoarguesian law and called them *generalised orthoarguesian laws.* In 2006, Mayet [4] described a family of equations $\mathcal{E}_A$, obtained with a technique similar to that used to derive the generalised orthoarguesian laws, that may further generalise these laws. In this paper, we obtain a result that opens a possibility that the latter class coincide with the former.

While the previous equations are not related to the states lattices admit, the other equations are. In 1981, Godowski [5] discovered an infinite family of equations derived by considering states on the lattice. In 1986, Mayet [6] generalised (strengthened) Godowski's equations with a new family, but the examples he gave were shown by Megill and Pavičić [3] to actually be instances of Godowski's equations. In 2006, Megill and Pavičić [7] showed the Mayet–Godowski class to be independent from the Godowski class. We also present a new algorithm for generating Mayet–Godowski equations (MGEs) that differs considerably from other methods described by Mayet [6,8].

In 2006, Mayet [4] discovered several new series of equations that hold provided the underlying field of $\mathcal{H}$ is real, complex, or quaternionic, which are also the ones of interest for quantum mechanics. Mayet found these by considering vector-valued states on $\mathcal{C}(\mathcal{H})$ and showed that they were independent of any of the other equations found so far. In this paper, we obtain several new results on these equations.

To achieve our results cited above, we developed several new algorithms. The main part of this paper describes the two most important ones, which are incorporated into the computer programs that found these results. The first algorithm (Sect. 4) determines whether a finite OML admits a "strong set of states" (defined elsewhere) and if not, an extension to the algorithm (Sect. 5) generates MGE that fails in the input OML but holds in every Hilbert lattice. The second algorithm (Sect. 6) enables us to prove whether or not this generated equation is independent from *every* equation in the infinite family found by Godowski. This second algorithm also enabled us to find Godowski lattices of much higher order than before and to show that it is possible to reduce their original size, therefore speeding up calculations that make use of them; some of these results are presented at the end of Sect. 6.

The last part of the paper presents two new results that were partly assisted by our programs. In Sect. 7, we show that an example provided by Mayet from his new family of orthoarguesian-related equations in fact can be derived from the generalised orthoarguesian laws, leaving open the problem of whether this new family has members that are strictly stronger than these laws. In Sect. 8, we show the solution to an open problem posed by Mayet [4] concerning his new families of equations related to strong sets of Hilbert space-valued states [8].

2. Definitions for Lattice Structures

We briefly recall the definitions we will need. For further information, see Refs. [3, 7, 9, 10].

Definition 2.1. [11] A *lattice* is an algebra $\mathrm{L} = \langle \mathcal{L}_{\mathrm{O}}, \cap, \cup \rangle$ such that the following conditions are satisfied for any $a, b, c \in \mathcal{L}_{\mathrm{O}}$: $a \cup b = b \cup a,\ a \cap b = b \cap a,\ (a \cup b) \cup c = a \cup (b \cup c),\ (a \cap b) \cap c = a \cap (b \cap c),\ a \cap (a \cup b) = a,\ a \cup (a \cap b) = a$.

Theorem 2.2. [11] *The binary relation* $\le$ *defined on* L *as* $a \le b \overset{\text{def}}{\Longleftrightarrow} a = a \cap b$ *is a partial ordering.*

Definition 2.3. [12] An *ortholattice* (OL) is an algebra $\langle \mathcal{L}_{\mathrm{O}}, {}', \cap, \cup, 0, 1 \rangle$ such that $\langle \mathcal{L}_{\mathrm{O}}, \cap, \cup \rangle$ is a lattice with unary operation $'$ called *orthocomplementation* which satisfies the following conditions for $a, b \in \mathcal{L}_{\mathrm{O}}$ (a' is called the *orthocomplement* of a): $a \cup a' = 1,\ a \cap a' = 0,\ a \le b\ \Rightarrow\ b' \le a',\ a'' = a$

Definition 2.4. [13,14] An OML is an OL in which the following condition holds: $a \leftrightarrow b = 1 \Leftrightarrow a = b$, where $a \leftrightarrow b = 1 \overset{\text{def}}{\Longleftrightarrow} a \to b = 1 \,\&\, b \to a = 1$, where $a \to b \overset{\text{def}}{=} a' \cup (a \cap b)$.

Definition 2.5. [15] We say that a and b *commute* in OML, and write aCb, when the following equation holds: $a \cap (a' \cup b) \le b$.

Definition 2.6. An OML which satisfies the following conditions is a *Hilbert lattice*, HL.[1]

1. *Completeness*: The meet and join of any subset of an HL exist.
2. *Atomicity*: Every non-zero element in an HL is greater than or equal to an atom. (An atom a is a non-zero lattice element with $0 < b \le a$ only if $b = a$.)
3. *Superposition principle*: (The atom c is a superposition of the atoms a and b if $c \ne a$, $c \ne b$, and $c \le a \cup b$.)
   (a) Given two different atoms a and b, there is at least one other atom c, $c \ne a$, and $c \ne b$, that is a superposition of a and b.
   (b) If the atom c is a superposition of distinct atoms a and b, then atom a is a superposition of atoms b and c.
4. *Minimum height*: The lattice contains at least two elements a, b satisfying: $0 < a < b < 1$.

[1] For additional definitions of the terms used in this section see Refs. [2,3,16,17].

Note that atoms correspond to pure states when defined on the lattice. We recall that the *irreducibility* and the *covering property* follow from the superposition principle [16, pp. 166, 167]. We also recall that any Hilbert lattice must contain a countably infinite number of atoms [18].

By Birkhoff's HSP theorem [19, p. 2], the family HL is not an equational variety, since a finite sublattice is not an HL. A goal of studying equations that hold in HL is to find the smallest variety that includes HL, so that the fewest number of non-equational (quantified) conditions such as the above will be needed to complete the specification of HL.

Definition 2.7. A *state* (also called *probability measures* or simply *probabilities* [17, 20–22]) on a lattice $\mathcal{L}$ is a function $m : \mathcal{L} \longrightarrow [0,1]$ such that $m(1) = 1$ and $a \perp b \Rightarrow m(a \cup b) = m(a) + m(b)$, where $a \perp b$ means $a \le b'$.

Lemma 2.8. *The following properties hold for any state* m:

$$m(a) + m(a') = 1, \tag{2.1}$$

$$a \le b \Rightarrow m(a) \le m(b), \tag{2.2}$$

$$0 \le m(a) \le 1, \tag{2.3}$$

$$m(a_1) = \cdots = m(a_n) = 1 \Leftrightarrow m(a_1) + \cdots + m(a_n) = n, \tag{2.4}$$

$$m(a_1 \cap \cdots \cap a_n) = 1 \Rightarrow m(a_1) = \cdots = m(a_n) = 1. \tag{2.5}$$

Definition 2.9. A set S of states on $\mathcal{L}$ is called a *strong*[2] set of states if

$$(\forall a, b \in \mathrm{L})([(\forall m \in S)(m(a) = 1 \Rightarrow m(b) = 1)] \Rightarrow a \le b). \tag{2.6}$$

Theorem 2.10. [3] *Every Hilbert lattice admits a strong set of states.*

3. Definitions of Equational Families Related to States

First, we will define the family of equations found by Godowski, introducing a special notation for them. These equations hold in any lattice admitting a strong set of states and thus, in particular, any Hilbert lattice [3].

Definition 3.1. Let us call the following expression the *Godowski identity*:

$$a_1 \overset{\gamma}{\equiv} a_n \overset{\text{def}}{=} (a_1 \to a_2) \cap (a_2 \to a_3) \cap \cdots \cap (a_{n-1} \to a_n) \cap (a_n \to a_1), \quad n = 3, 4, \ldots \tag{3.1}$$

We define $a_n \overset{\gamma}{\equiv} a_1$ in the same way with variables a_i and a_{n-i+1} swapped.

[2] Some authors use the term *rich* instead of *strong*, e.g. Ref. [23, p. 21].

Theorem 3.2. *Godowski's equations* [5]

$$a_1 \stackrel{\gamma}{\equiv} a_3 \;=\; a_3 \stackrel{\gamma}{\equiv} a_1, \tag{3.2}$$
$$a_1 \stackrel{\gamma}{\equiv} a_4 \;=\; a_4 \stackrel{\gamma}{\equiv} a_1, \tag{3.3}$$
$$a_1 \stackrel{\gamma}{\equiv} a_5 \;=\; a_5 \stackrel{\gamma}{\equiv} a_1, \tag{3.4}$$
$$\dots$$

hold in all ortholattices, OL*s, with strong sets of states. An* OL *to which these equations are added is a variety smaller than* OML*.*

We shall call these equations n-Go (3-Go, 4-Go, etc.). We also denote by nGO (3GO, 4GO, etc.) the OL variety determined by n-Go, and we call equation n-Go the nGO law.[3]

Next, we define a generalisation of this family, first described by Mayet [24]. These equations also hold in all lattices admitting a strong set of states, and in particular in all HLs.

Definition 3.3. An MGE is an equality with $n \ge 2$ conjuncts on each side:

$$t_1 \cap \cdots \cap t_n = u_1 \cap \cdots \cap u_n, \tag{3.5}$$

where each conjunct t_i (or u_i) is a term consisting of either a variable or a disjunction of two or more distinct variables:

$$t_i = a_{i,1} \cup \cdots \cup a_{i,p_i}, \quad \text{i.e. } p_i \text{ disjuncts,} \tag{3.6}$$
$$u_i = b_{i,1} \cup \cdots \cup b_{i,q_i}, \quad \text{i.e. } q_i \text{ disjuncts} \tag{3.7}$$

and where the following conditions are imposed on the set of variables in the equation:

1. All variables in a given term t_i or u_i are mutually orthogonal.
2. Each variable occurs the same number of times on each side of the equality.

We will call a lattice in which all MGEs hold an MGO; i.e. MGO is the largest class of lattices (equational variety) in which all MGEs hold.

The following three theorems about MGEs and MGOs are proved in Ref. [7].

Theorem 3.4. *Every* MGE *holds in any ortholattice $\mathcal{L}$ admitting a strong set of states and thus, in particular, in any Hilbert lattice.*

Theorem 3.5. *The family of all* MGEs *includes, in particular, the Godowski equations* [Eqs. (3.2), (3.3),...]; *in other words, the class* MGO *is included in* nGO *for all* n*.*

[3] The equation n-Go can also be expressed with $2n$ variables: $a_1 \perp b_1 \perp a_2 \perp b_2 \perp \cdots a_n \perp b_n \perp a_1 \Rightarrow (a_1 \cup b_1) \cap \cdots (a_n \cup b_n) \le b_1 \cup a_2$, where $n \ge 3$. We remark that if we set $n = 2$, this equation holds in all OMLs, answering a question in Ref. [8, p. 536]. This can be seen as follows. The equation that results from setting $n = 2$ in the equation series of Theorem 3.2 has two variables and is easily shown to hold in all OMLs. The proof of Th. 3.19 of Ref. [3], which converts it to the $2n$-variable form, involves only OML manipulations.

Theorem 3.6. *The class* MGO *is properly included in all* nGO*s, i.e. not all* MGEs *can be deduced from the equations* n-Go.

Definition 3.7. A *condensed state equation* is an abbreviated representation of an MGE constructed as follows: all (orthogonality) hypotheses are discarded, all meet symbols, $\cap$, are changed to $+$, and all join symbols, $\cup$, are changed to juxtaposition.

For example, the 3-Go equation can be expressed as [7]:

$$\begin{aligned} & a \perp d \perp b \perp e \perp c \perp f \perp a \\ & \quad \Rightarrow (a \cup d) \cap (b \cup e) \cap (c \cup f) = (d \cup b) \cap (e \cup c) \cap (f \cup a), \end{aligned} \tag{3.8}$$

which, in turn, can be expressed by the condensed state equation

$$ad + be + cf = db + ec + fa. \tag{3.9}$$

The one-to-one correspondence between these two representations of an MGE should be obvious.

4. Finding States on Finite Lattices

It is possible to express the set of constraints imposed by states as a linear programming (LP) problem. LP is used by industry to minimise cost, labour, etc., and many efficient programs have been developed to solve these problems, most of them based on the simplex algorithm.

We will examine a particular example in detail to illustrate how the problem is expressed. For this example, we will consider a Greechie diagram with 3-atom blocks, although the principle is easily extended to any number of blocks.

If m is a state, then each 3-atom block with atoms (a, b, c) and complements (a', b', c') imposes the following constraints:

$$\begin{aligned} m(a) + m(b) + m(c) &= 1 \\ m(a') + m(a) &= 1, \\ m(b') + m(b) &= 1, \\ m(c') + m(c) &= 1, \\ m(x) &\geq 0, \quad x = a, b, c, a', b', c'. \end{aligned} \tag{4.1}$$

To obtain Eq. (4.1), note that in any Boolean block, $a \perp b \perp c \perp a$, so $m(a) = 1 - m(a') = 1 - m(b \cup c) = 1 - m(b) - m(c)$.

Let us take the specific example of the Peterson lattice, which we know does not admit a set of strong states. The Greechie diagram for this lattice, shown in Fig. 1, can be expressed with the textual notation

`123,345,567,789,9AB,BC1,2E8,4FA,6DC,DEF.`

(see Ref. [7]), where each digit or letter represents an atom, and groups of them represent blocks (edges of the Greechie diagram).

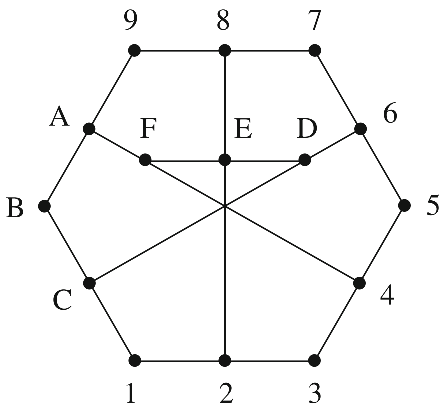

FIGURE 1. Greechie diagram for the Peterson lattice.

Referring to the textual notation, we designate the atoms by $1, 2, \ldots, F$ and their orthocomplements by $1', 2', \ldots, F'$. We will represent the values of state m on the atoms by $m(1), m(2), \ldots, m(F)$. This gives us the following constraints for the 10 blocks:

$$\begin{aligned}
m(1) + m(2) + m(3) &= 1,\\
m(3) + m(4) + m(5) &= 1,\\
m(5) + m(6) + m(7) &= 1,\\
m(7) + m(8) + m(9) &= 1,\\
m(9) + m(A) + m(B) &= 1,\\
m(B) + m(C) + m(1) &= 1,\\
m(2) + m(E) + m(8) &= 1,\\
m(4) + m(F) + m(A) &= 1,\\
m(6) + m(D) + m(C) &= 1,\\
m(D) + m(E) + m(F) &= 1.
\end{aligned}$$

In addition, we have $m(a') + m(a) = 1$, $m(a) \geq 0$, and $m(a') \geq 0$ for each atom a, adding potentially an additional $15 \times 3 = 45$ constraints. However, we can omit all but one of these since most orthocomplemented atoms are not involved in this problem, the given constraints are sufficient to ensure that the state values for atoms are less than 1, and the particular LP algorithm we used assumes all variables are nonnegative. This speeds up the computation considerably. The only one we will need is $m(7) + m(7') = 1$ because, as we will see, the orthocomplemented atom $7'$ will be part of the full problem statement.

We pick two incomparable nodes, 1 and $7'$, which are on opposite sides of the Peterson lattice. (The program will try all possible pairs of incomparable nodes, but for this example, we have selected a priori a pair that will provide us with the answer). Therefore, it is the case that $\sim 1 \leq 7'$. If the Peterson lattice admitted a strong set of states, for any state m we would have:

$$(m(1) = 1 \Rightarrow m(7') = 1) \Rightarrow 1 \leq 7'.$$

Since the conclusion is false, for some m we must have

$$\begin{aligned} &\sim (m(1) = 1 \Rightarrow m(7') = 1), \\ &\text{i.e.} \sim (\sim m(1) = 1 \vee m(7') = 1), \\ &\text{i.e. } m(1) = 1 \ \& \sim m(7') = 1. \end{aligned}$$

So this gives us another constraint:

$$m(1) = 1;$$

and for a set of strong states to exist, there must be some m such that

$$m(7') < 1.$$

So, our final LP problem becomes (expressed in the notation of the publicly available program `lp_solve`[4]):

```
min: m7′;
m1 = 1;
m7 + m7′ = 1;
m1 + m2 + m3 = 1;
m3 + m4 + m5 = 1;
m5 + m6 + m7 = 1;
m7 + m8 + m9 = 1;
m9 + mA + mB = 1;
mB + mC + m1 = 1;
m2 + mE + m8 = 1;
m4 + mF + mA = 1;
m6 + mD + mC = 1;
mD + mE + mF = 1;
```

which means "minimise $m(7')$, subject to constraints $m(1) = 1, m(7) + m(7') = 1, \ldots$." The variable to be minimised, $m(7')$, is called the *objective function* (or "cost function"). When this problem is given to `lp_solve`, it returns an objective function value of 1. This means that regardless of m, the other constraints impose a minimum value of 1 on $m(7')$, contradicting the requirement that $m(7') < 1$. Therefore, we have a proof that the Peterson lattice does not admit a set of strong states.

The program `states.c` that we use reads a list of Greechie diagrams and, for each one, indicates whether or not it admits a strong set of states. The program embeds the `lp_solve` algorithm, wrapping around it an interface that translates, internally, each Greechie diagram into the corresponding LP problem.

[4] Version 3.2, available at http://m3k.grad.hr/lp_solve_32/ (as of January 2010).

5. Generation of MGEs from Finite Lattices

When the LP problem in the previous section finds a pair of incomparable nodes that prove that the lattice admits no strong set of states, the information in the problem can be used to find an equation that holds in any OML admitting a strong set of states, and in particular in HL, but fails in the OML under test. Typically, an OML to be tested was chosen because it does not violate any other known HL equation. Thus, by showing an HL equation that fails in the OML under test, we would have found a new equation that holds in HL and is independent from other known equations.

The set of constraints that lead to the objective function value of 1 in our LP problem turns out to be redundant. Our algorithm will try to find a minimal set of hypotheses (constraints) that are needed. The equation-finding mode of the `states.c` program incorporates this algorithm, which will try to weaken the constraints of the LP problem one at a time, as long as the objective function value remains 1 (as in the problem in the previous section). The equation will be constructed based on a minimal set of unweakened constraints that results.

The theoretical basis for the construction is described in the proof of Theorem 30 of Ref. [7]. Here, we will describe the algorithm by working through a detailed example.

Continuing from the final LP problem of the previous section, the program will test each constraint corresponding to a Greechie diagram block, i.e. each equation with three terms, as follows. It will change the right-hand side (r.h.s.) of the constraint equation from $= 1$ to ≤ 1, thus weakening it, then it will run the LP algorithm again. If the weakened constraint results in an objective function value $m(7') < 1$, it means that the constraint is needed to prove that the lattice does not admit a strong set of states, so we restore the r.h.s. of that constraint equation back to 1. On the other hand, if the objective function value remains $m(7') = 1$ (as in the original problem), a tight constraint on that block is not needed for the proof that the lattice does not admit a strong set of states, so we leave the r.h.s. of that constraint equation at ≤ 1.

After the program completes this process, the LP problem for this example will look like this:

```
min: m7′;
m1 = 1;
m7 + m7′ = 1;
m1 + m2 + m3 <= 1;
m3 + m4 + m5 = 1;
m5 + m6 + m7 <= 1;
m7 + m8 + m9 <= 1;
m9 + mA + mB = 1;
mB + mC + m1 <= 1;
m2 + mE + m8 = 1;
m4 + mF + mA <= 1;
```

```
m6 + mD + mC = 1;
mD + mE + mF <= 1;
```

Six out of the 10 blocks have been made weaker, and the LP algorithm will show that the objective function has remained at 1. We now have enough information to construct the MGE, which we will work within the abbreviated form of a condensed state equation (Definition 3.7).

1. Since $m(1) = 1$, the other atoms in the two blocks (three-term equations) using it will be 0. Thus $m(2) = m(3) = m(B) = m(C) = 0$.
2. For each of the four blocks that have $= 1$ on the r.h.s., we suppress the atoms that are 0 and juxtapose the remaining two atoms in each block. For example, in $m(3) + m(4) + m(5) = 1$, we ignore $m(3) = 0$, and collect the atoms from the remaining two terms to result in 45 (4 juxtaposed with 5). Then we join all four pairs with + to build the left-hand side (l.h.s.) form for the condensed state equation:
$$45 + 9A + E8 + 6D. \tag{5.1}$$
3. For the r.h.s. of the equation, we scan the blocks with weakened constraints. From each block, we pick out and juxtapose those atoms that also appear on the l.h.s. and discard others. For example, in $m(5) + m(6) + m(7) \leq 1$, 5 and 6 appear in Eq. (5.1) but 7 does not. Joining the juxtaposed groups with +, we build the r.h.s.:
$$56 + 89 + 4A + DE.$$
Note that out of six weakened constraints, two of them have no atoms at all in common with Eq. (5.1) and are therefore ignored.
4. Equating the two sides, we obtain the form of the condensed state equation:
$$45 + 9A + E8 + 6D = 56 + 89 + 4A + DE.$$
5. Replacing the atoms with variables, the final condensed state equation becomes:
$$ab + cd + ef + gh = bg + fc + ad + he. \tag{5.2}$$
6. Finally, the number of occurrences of each variable on must match with each side of the condensed state equation. In this example, that is already the case. But in general, there may be terms that will have to be repeated in order to make the numbers balance. An example with such "degenerate" terms is shown as Eq. (47) of Ref. [7].

Equation (5.2) will be recognised, after converting it to an MGE, as the 4-Go equation, which as is well-known holds in all OMLs that admit a strong set of states but fails in the Peterson lattice (Fig. 1) [3].

Remark. We emphasise that the above algorithm is essentially heuristic, in that its purpose is to make use of the existing functions in the `states.c` program to assist producing new equations with less manual labour. In particular, there is no guarantee that it will produce the strongest equation possible that is deducible from an OML, nor even that it will be able to find an equation at all. Indeed, a few pathological OMLs have been found where the algorithm does not find terms that can be balanced (in the sense mentioned in the last item above).

While further refinement of the algorithm may be possible, the point of it is to provide a practical method to quickly generate new equations for further study. These can be independently verified as both holding in every OML with a strong set of states while at the same time being stronger than any equation known up to that point. From a practical standpoint, at this stage we are mainly interested in studying small equations with few variables, even if they are not the strongest possible that can be generated from an OML, simply because they are more tractable to work with. In a similar fashion, many of our results for n-Go and nOA equations were obtained by first studying them for small n. Of course, our goals may change once we gain a better understanding of the general properties of MGEs and how they can be classified.

6. Checking n-Go Equations on Finite Lattices

For the general-purpose checking of whether an equation holds in a finite lattice, the authors have primarily used a specialised program, `latticeg.c`, that tests an equation provided by the user against a list of Greechie diagrams (OMLs) provided by the user. This program has been described in Ref. [25]. While it has proved essential to our work, a drawback is that the run time increases quickly with the number of variables in and size of the input equation, making it impractical for huge equations.

But there is another limitation in principle, not just in practice, for the use of the `latticeg.c` program. In our work with MGEs, we are particularly interested in those lattices having no strong set of states but on which all of the successively stronger n-Gos pass, for all n less than infinity. This would prove that any MGE failing in that lattice is independent from all n-Gos and thus represents a new result. The `latticeg.c` program can, of course, check only a finite number of such equations, and when n becomes large, the program is too slow to be practical. And in any case, it cannot provide a proof, but only evidence, that a particular lattice does not violate n-Go for any n.

Both of these limitations are overcome by a remarkable algorithm based on dynamic programming, that was suggested by Brendan McKay. This algorithm was incorporated into a program, `latticego.c`, that is run against a set of lattices. No equation is given to the program; instead, the program tells the user the first n for which n-Go fails or whether it passes for all n. The program runs very quickly, depending only on the size of the input lattice, with a run time proportional to the fourth power of the lattice size (number of nodes) m, rather than increasing exponentially with the equation size (number of variables) n as with the `latticeg.c` program that checks against arbitrary equations.

To illustrate the algorithm, we will consider the specific case of 7-Go. From this example, the algorithm for the general case of n-Go will be apparent. We consider 7-Go written in the following equivalent form [3]:

$$\begin{aligned}(a_1 \to a_2) \cap (a_2 \to a_3) \cap (a_3 \to a_4) \cap (a_4 \to a_5) \\ \cap (a_5 \to a_6) \cap (a_6 \to a_7) \cap (a_7 \to a_1) \le a_1 \to a_7. \end{aligned} \tag{6.1}$$

We define intermediate "operations" $E_1, \ldots, E_6$ along with a predicate which provides the answer:

$$\begin{aligned}
E_1(a_1, a_2) &= a_1 \to a_2, \\
E_2(a_1, a_2, a_3) &= E_1(a_1, a_2) \cap (a_2 \to a_3), \\
E_3(a_1, a_2, a_3, a_4) &= E_2(a_1, a_2, a_3) \cap (a_3 \to a_4), \\
E_4(a_1, a_2, a_3, a_4, a_5) &= E_3(a_1, a_2, a_3, a_4) \cap (a_4 \to a_5), \\
E_5(a_1, a_2, a_3, a_4, a_5, a_6) &= E_4(a_1, a_2, a_3, a_4, a_5) \cap (a_5 \to a_6), \\
E_6(a_1, a_2, a_3, a_4, a_5, a_6, a_7) &= E_5(a_1, a_2, a_3, a_4, a_5, a_6) \cap (a_6 \to a_7), \\
\text{answer}(a_1, a_7) &= (E_6(a_1, a_2, a_3, a_4, a_5, a_6, a_7) \cap (a_7 \to a_1)) \le (a_1 \to a_7).
\end{aligned}$$

Sets of values $V_2, \ldots, V_6$ are computed as follows:

$$\begin{aligned}
&V_2(a_1, a_3) = \{E_2(a_1, a_2, a_3) | a_2\}, \\
&V_3(a_1, a_4) = \{E_3(a_1, a_2, a_3, a_4) | a_2, a_3\}, \\
&V_4(a_1, a_5) = \{E_4(a_1, a_2, a_3, a_4, a_5) | a_2, a_3, a_4\}, \\
&V_5(a_1, a_6) = \{E_5(a_1, a_2, a_3, a_4, a_5, a_6) | a_2, a_3, a_4, a_5\}, \\
&V_6(a_1, a_7) = \{E_6(a_1, a_2, a_3, a_4, a_5, a_6, a_7) | a_2, a_3, a_4, a_5, a_6\}, \\
&\text{For all } a_1, a_7 : \text{answer}(a_1, a_7) \text{ follows from } V_6(a_1, a_7), a_7 \to a_1, \\
&\qquad\qquad \text{and } a_1 \to a_7.
\end{aligned}$$

For example, $V_4(a_1, a_5)$ is the set of values $E_4(a_1, a_2, a_3, a_4, a_5)$ can have when a_2, a_3, a_4 range over all possibilities. If answer(a_1, a_7) is true for all possible a_1 and a_7, then 7-Go holds in the lattice, otherwise it fails.

The computation time is estimated as follows, where m is the number of nodes in the test lattice:

$$\begin{aligned}
&\text{Each } V_2(a_1, a_3) \text{ can be found in } O(m) \text{ time; } O(m^3) \text{ total}, \\
&\text{Each } V_3(a_1, a_4) \text{ can be found in } O(m^2) \text{ time from } V_2; O(m^4) \text{ total}, \\
&\text{Each } V_4(a_1, a_5) \text{ can be found in } O(m^2) \text{ time from } V_3; O(m^4) \text{ total}. \\
&\qquad\qquad \vdots
\end{aligned}$$

So the total time is $O(m^4)$, when the algorithm is applied to a specific n-Go equation. If it were not for the typical convergent behaviour described in the following, we would also multiply this time by $n - 2$, i.e. the number of passes $V_2, \ldots, V_{n-1}$. In fact, only in rare cases do we require a computation of V_i for i greater 10 or so, as we will explain.

The program is written so that it only has to compute additional "inner terms" to process the next n-Go equation. Remarkably, when a lattice does not violate any n-Go, our observation has been that the addition of new terms almost always converges to a fixed value rather quickly, meaning that V_n for $(n+1)$-Go remains the same as V_{n-1} for n-Go. This almost always happens for $n < 10$, and when it does, we can terminate the algorithm and say with certainty that no further increase in n will cause an n-Go equation to fail in the lattice. (If it does not happen, the program will tell us that, but such a case has so far not been observed. The program has an arbitrary cutoff point of $n = 100$, after which the algorithm will terminate. All of our observed runs have always either converged or failed far below this point, and in any case the cutoff can be increased with a parameter setting.) Convergence provides a proof that the entire class of Godowski equations (for *all* $n < \infty$) will pass in the lattice. Such a feat is not possible with ordinary lattice-checking programs, since an infinite number of equations would have to be tested.

At this point, we do not have a good explanation for this quickly convergent behaviour. It is simply an empirical observation.

When a lattice does violate some n-Go, that result tends to be found even faster: the algorithm terminates, and the program tells us the first n at which an n-Go equation fails in the lattice. Since n-Go can be derived from $n+1$-Go, failure is also implied for all greater n. The algorithm can also be used for a secondary purpose: it can scan a collection of lattices to determine efficiently which of them satisfy n-Go but violate $(n+1)$-Go and find the smallest ones with this property.

We caution the reader that m above represents lattice nodes, not Greechie diagram atoms. For a fixed block size of say 3, which is the most common one we have used, these numbers are proportional. However, the number of nodes in a block (Boolean algebra) grows exponentially with the number of atoms in the block.

Here, we should explain why we use our algorithm to find n at which an n-Go equation fails in the lattice when it is well known [5] that n-Go fails in Godowski's "wagon wheel" of order n which can be easily constructed for each n. The answer is that smaller lattices significantly reduce the run time needed to check an equation conjectured to be equivalent to an n-Go. The $(n+1)$st wagon wheel lattice has six atoms more than the nth one, and we have shown in Ref. [3] that our algorithms give the smallest lattices in which 4-Go to 7-Go fail that are on average only 2.7 atoms apart. Our most recent computations[5] presented below show that for higher ns, this number can be still smaller and as we see from Figs. 2 and 3 for, e.g. 9-Go to 12-Go it is on average 1.

In our textual notation, the OMLs from Fig. 2 read:

(a) 123, 345, 567, 789, 9AB, BCD, DEF, FGH, HI1, 2NE, 4JD, 6KC, IJ8, GL9, NMA, 2LK.
(b) 123, 345, 567, 789, 9AB, BCD, DEF, FGH, HIJ, JKL, LMN, NO1, KP8, 4PG, JQA, OE6, KD2, MC5.

[5] For this purpose, we used the `latticego` algorithm together with the lattice-generating program `gengre` described in Ref. [25] and extended in [26].

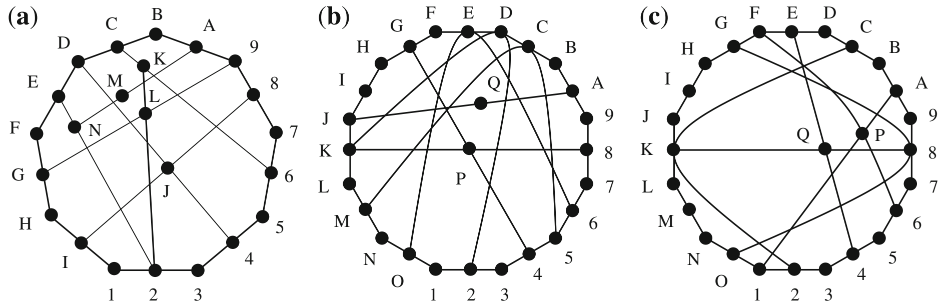

FIGURE 2. **a** 23-16-p7go-f8go-a—one of two smallest lattices that pass 7-Go and violate 8-Go; **b** 26-18-p8go-f9go-a—one of 23 smallest; **c** 26-18-p9go-f10go-a—one of 42 smallest.

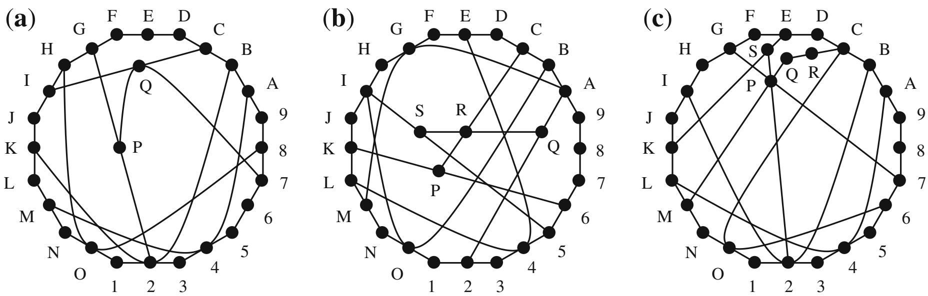

FIGURE 3. **a** 26-18-p9go-f10go-b that also violates E_3, Eq. (8.2), from Sect. 8; **b** 28-20-p10go-f11go-a—one of the over 50 smallest OMLs that pass 10-Go and violate 11-Go—it also violates E_3; **c** 28-20-p11go-f12go-a—one of the over 50 smallest OMLs that pass 11-Go and violate 12-Go—it passes E_3.

(c) 123, 345, 567, 789, 9AB, BCD, DEF, FGH, HIJ, JKL, LMN, NO1, KQ8, 4QE, 1PA, 6PF, O8G, 2KC.

The smallest lattice that satisfies 6-Go and violates 7-Go, 24-16-p6go-f7go (24 atoms, 16 blocks) [3], is bigger than the smallest one that satisfies 7-Go and violates 8-Go, 23-16-p7go-f8go (23 atoms, 16 blocks) shown in Fig. 2. Also, 26-18-p8go-f9go-a and 26-18-p9go-f10go-a are of the same size.

In the textual notation, the OMLs from Fig. 3a–c read:

123, 345, 567, 789, 9AB, BCD, DEF, FGH, HIJ, JKL, LMN, NO1, 2PG, IQC, 7QP, HO8, K2B, M4A.
123,345,567,789,9AB,BCD,DEF,FGH,HIJ,JKL,LMN,NO1,MGA,IOB,L4E,KP6,IS5,2QA,PRC,QSR.
123,345,567,789,9AB,BCD,DEF,FGH,HIJ,JKL,LMN,NO1,CO6,I2B,L4A,KSE,MPQ,QRC,2PS,7PG.

We see that 28-20-p10go-f11go-a and 28-20-p11go-f12go-a are of the same size and contain only two more atoms than 26-18-p9go-f10go-a,b.

7. Can Generalised Orthoaguesian Equations be Enlarged?

There is an infinite series of algebraic OML equations that are apparently at least partly independent of the conditions that the superpositions and the states' OMLs admit impose on OMLs. In particular, the series properly overlaps with those characterising states and superpositions, that have to hold in any Hilbert lattice characterising quantum systems. A class of such equations are the so-called *generalised orthoarguesian equations* nOA discovered by Megill and Pavičić [3,7]. We introduce them as follows.

Definition 7.1. We define an operation $\stackrel{(n)}{\equiv}$ on n variables $a_1,\dots,a_n$ $(n \geq 3)$ as follows:

$$a_1 \stackrel{(3)}{\equiv} a_2 \stackrel{\mathrm{def}}{=} ((a_1 \to a_3) \cap (a_2 \to a_3)) \cup ((a_1' \to a_3) \cap (a_2' \to a_3)), \tag{7.1}$$

$$a_1 \stackrel{(n)}{\equiv} a_2 \stackrel{\mathrm{def}}{=} (a_1 \stackrel{(n-1)}{\equiv} a_2) \cup ((a_1 \stackrel{(n-1)}{\equiv} a_n) \cap (a_2 \stackrel{(n-1)}{\equiv} a_n)), \quad n \geq 4. \tag{7.2}$$

Theorem 7.2. *The* nOA *laws*

$$(a_1 \to a_3) \cap (a_1 \stackrel{(n)}{\equiv} a_2) \leq a_2 \to a_3 \tag{7.3}$$

hold in any HL.

In Ref. [8, p. 530], Mayet attempted to enlarge the generalised orthoarguesian laws by deriving a family of equations, $\mathcal{E}_A$, which includes the nOA laws as a special case. The equations $\mathcal{E}_A$ are shown to hold in all HLs using a similar method of proof used to obtain the nOA laws. In particular, subset relations between subspace sums in a Hilbert space $\mathcal{H}$ are found by considering sums and differences of their member vectors. Subspace sums are then converted to closed subspace joins, using either the relation $\mathcal{H}_A + \mathcal{H}_B \subseteq \mathcal{H}_A \cup \mathcal{H}_B$, which holds in general for subspaces $\mathcal{H}_A$ and $\mathcal{H}_B$, or the equation $\mathcal{H}_A + \mathcal{H}_B = \mathcal{H}_A \cup \mathcal{H}_B$, which holds whenever $\mathcal{H}_A$ and $\mathcal{H}_B$ are orthogonal.

Mayet gives an example from family $\mathcal{E}_A$ that is at least as strong as the 3OA law[6] in the sense that it implies the latter. Letting $t_1 = c \cup ((a \cup d \cup e) \cap (b \cup c))$, $t_2 = d \cup ((a \cup c \cup e) \cap (b \cup d))$, and $t_3 = e \cup ((a \cup c \cup d) \cap (b \cup e))$, Mayet obtains the following equation:[7] [8, p. 531]

$$\begin{aligned} &a \perp b \quad \& \quad c \perp d \quad \& \quad d \perp e \quad \& \quad c \perp e \\ &\qquad \Rightarrow (a \cup b) \cap (c \cup d \cup e) \leq a \cup (b \cap t_3 \cap t_2 \cap t_1). \end{aligned} \tag{7.4}$$

By setting $e = 0$ in Eq. (7.4), we obtain an equation that obviously implies the 3OA law, which can be seen when the 3OA law is expressed in the following four-variable form [3, Theorem 4.9]:

$$a \perp b \;\; \& \;\; c \perp d \Rightarrow (a \cup b) \cap (c \cup d) \leq a \cup (b \cap (c \cup ((a \cup d) \cap (b \cup c)))). \tag{7.5}$$

[6] Mayet uses the notation OA_{n-2} for the equation that we call the nOA law.

[7] Although condition (7.4) has hypotheses, it is equivalent to a (closed) equation by [6, p. 168, Lemma 1]. Therefore, we are justified calling it an equation, which we will do for this and similar conditions.

However, as we prove in the following theorem, it turns out that the 3OA law Eq. (7.5) also implies Eq. (7.4), meaning that the latter is not *strictly* stronger than the former; in other words, they are equivalent.

Theorem 7.3. *An* OML *in which Eq.* (7.4) *holds is a* 3OA *and vice-versa.*

Proof. As we just described, Eq. (7.4) implies the implies the 3OA law. For the converse, assume that we are given the 3OA law and that the hypotheses of Eq. (7.4) hold. We obtain three substitution instances of the 3OA law by putting $d\cup e$ for d; then d, $c\cup e$ for c, d; then e, $c\cup d$ for c, d in Eq. (7.5). The hypotheses of Eq. (7.5) are then satisfied, and we conclude, respectively,

$$\begin{aligned}
(a\cup b)\cap(c\cup d\cup e) &\le a\cup(b\cap(c\cup((a\cup d\cup e)\cap(b\cup c)))) = a\cup(b\cap t_1),\\
(a\cup b)\cap(d\cup c\cup e) &\le a\cup(b\cap(d\cup((a\cup c\cup e)\cap(b\cup d)))) = a\cup(b\cap t_2),\\
(a\cup b)\cap(e\cup c\cup d) &\le a\cup(b\cap(e\cup((a\cup c\cup d)\cap(b\cup e)))) = a\cup(b\cap t_3).
\end{aligned}$$

Conjoining the r.h.s.,

$$(a\cup b)\cap(c\cup d\cup e) \le (a\cup(b\cap t_1))\cap(a\cup(b\cap t_2))\cap(a\cup(b\cap t_3)).$$

Since $a\perp b$, we have that a commutes with $b\cap t_i$, $i=1,2,3$, so we can use the Foulis–Holland theorem (F-H; see, e.g. [20, p. 25]) to apply the distributive law to the r.h.s.:

$$\begin{aligned}
(a\cup b)\cap(c\cup d\cup e) &\le (a\cup((b\cap t_1)\cap(b\cap t_2)\cap(b\cap t_3))\\
&= (a\cup(b\cap t_1\cap t_2\cap t_3),
\end{aligned}$$

which is the conclusion of Eq. (7.4) as required. □

In a word, whether the $\mathcal{E}_A$ equations strictly include the nOA equations or coincide with them remains an open problem.

8. Mayet's E-equations and a Solution to a Related Open Problem

A third class of equations makes use of "vector measures" [27]. Several new families of equations based on them were found by Mayet [4]. He called these measures *Hilbert space-valued states*, defined as follows.

Definition 8.1. A *real Hilbert space-valued state*—we call it an $\mathcal{RH}$ *state*—on an OML $\mathcal{L}$ is a function $s:\mathcal{L}\longrightarrow\mathcal{RH}$, where $\mathcal{RH}$ is a Hilbert space defined over a real field, such that (a) $||s(1_{\mathcal{L}})||=1$, where $s(a)\in\mathcal{RH}$ is a state vector, $||s(a)||=\sqrt{(s(a),s(a))}$ is the Hilbert space norm, and $a\in\mathcal{L}$; (b) $(\forall a,b\in\mathcal{L})\,[\,a\perp b\Rightarrow s(a\cup b)=s(a)+s(b)\,]$, where $a\perp b$ means $a\le b'$; (c) $(\forall a,b\in\mathcal{L})\,[\,a\perp b\Rightarrow s(a)\perp s(b)\,]$, where $s(a)\perp s(b)$ means the inner product $(s(a), s(b))=0$.

We also define a subclass of HL for which Definition 8.1 will later become relevant:

Definition 8.2. A *quantum*[8] *Hilbert lattice*, QHL, is a Hilbert lattice orthoisomorphic to the set of closed subspaces of the Hilbert space defined over either a real field, or a complex field, or a quaternion skew field.

The conditions of Lemma 2.8 hold when we replace a real state value $m(a)$ with the square of the norm of the $\mathcal{RH}$ state value $s(a)$. In addition, there are number of properties that hold for $\mathcal{RH}$ states—see [4,7]. The following definition of a strong set of $\mathcal{RH}$ states closely follows Definition 2.9, with an essential difference in the range of the states. We also define a *strong set of* $\mathcal{CH}$ and a $\mathcal{QH}$ states.

Definition 8.3. A set S of $\mathcal{RH}$ states $s : \mathcal{L} \longrightarrow \mathcal{RH}$ is called a *strong set of* $\mathcal{RH}$ states if

$$(\forall a, b \in \mathrm{L})([(\forall s \in S)(||s(a)|| = 1 \Rightarrow ||s(b)|| = 1)] \Rightarrow a \le b). \tag{8.1}$$

Theorem 8.4. [4], [7, p. 784] *Any* QHL *admits a strong set of* $\mathcal{RH}$ *states.*

Mayet derives three new families of equations, E_n, E_n^*, and E_n', which hold in all HLs but do not (for $n \ge 3$) hold in all OMLs not admitting strong sets of Hilbert space-valued states [8]. For variables $a_1, \dots, a_n, b_1, \dots, b_n, r$, let (Ω) be the set of conditions $a_i \perp a_j$ $(i \ne j)$ and $a_i \perp b_i$, for $1 \le i, j \le n$. Define $a = a_1 \cup \dots \cup a_n$, $b = b_1 \cup \dots \cup b_n$, and $q = (a_1 \cup b_1) \cap \dots \cap (a_n \cup b_n)$. Equations E_n, E_n^*, and E_n' $(n \ge 2)$ are defined as

$$(\Omega) \Rightarrow a \cap q \le b, \tag{8.2}$$

$$((\Omega) \ \ \& \ \ r \perp a) \Rightarrow (a \cup r) \cap q \le b \cup r, \tag{8.3}$$

$$((\Omega) \ \ \& \ \ r \perp a) \Rightarrow q \cap (q \to r') \cap (a \cup r) \le b, \tag{8.4}$$

respectively.

Theorem 8.5. *Equations* E_n *and* E_n^* *fail in* $\mathcal{L}_n$ *given in Fig.* 1 *of* [4], $n \ge 3$. *Equation* E_n' *fails in* $\mathcal{L}_n'$ *given in Fig.* 5 *of* [4], $n \ge 3$. *All of these equations hold in any* OML *with a strong set of* $\mathcal{RH}$ *states.*

Proof. See Ref. [4] or Ref. [7, p. 785] for E_n, and Ref. [4] for E_n^* and E_n'. (Th. 36 in [7] should be corrected to read "$\mathcal{L}_n$" in place of "$\mathcal{L}_i$, $i = 1, \dots, n$.") □

The equations of Theorem 8.5, which hold in every QHL, do not hold in every HL. They are independent of the modular law and of any nOA law, nGO law, MGE, or combination of them added to the axioms for OML [4], [7, p. 786].

[8] Mayet [4] calls our family QHL by the name *classical Hilbert lattices*, but since the real and complex fields as well as the quaternion skew field over which the corresponding Hilbert space is defined are characteristic of its application in quantum mechanics, one of us (MP) prefers to call these lattices quantum. Mayet uses the notation HL for the class we call QHL. He also uses the notation generalised Hilbert lattices (GHL) for the larger class defined by omitting the field requirement in Definition 8.2, which (see Ref. [28, Sects. 33, 34]) is equal to the family we call HL in Definition 2.6. Finally, the notation $\mathcal{C}(\mathcal{H})$ is often used to specify the lattice of closed subspaces of a particular Hilbert space $\mathcal{H}$, typically when its underlying field is complex (e.g. Ref. [20, p. 64]), in which case $\mathcal{C}(\mathcal{H}) \in$ QHL.

As an example, that E_3 and E_3^* are independent of the nOA laws for $n = 3, 4, 5$ is shown by the fact that OML L42 (Fig. 7 (b) from [3]) satisfies the latter equations but violates E_3 and E_3^*. Also, our `states` program shows that L42 has a strong set of states and thus satisfies all n-Go and MGE equations, showing the independence of E_3 and E_3^* from these. L42 is the smallest lattice with these properties. Equation E_3 does not fail in lattice $\mathcal{L}_3$ given in Fig. 5 of [4], showing that E_3' is strictly stronger than E_3.

Another example is given in Figs. 2 and 3. OMLs 26-18-p9go-f10go-b and 28-20-p10go-f11go-a violate E_3, while 23-16-p7go-f8go-a, 26-18-p8go-f9go-a, 26-18-p9go-f10go-a, and 28-20-p11go-f12go-a satisfy it. All of them satisfy E_4. This OMLs are the only known test on E_3 and E_4 apart from those mentioned above.

Yet another example is the following Mayet's OML (30 atoms, 19 blocks)

123,456,789,ABC,DEF,GHI,JKL,MNO,PQR,STU,147S,ADGT,JMPU,3CL,6FO,9IR,2EQ,5HK,8BN.

(see Fig. 1 of [29]), which satisfies both E_3 and E_4 as well as all n-Go but does not admit any state.

The E_n, E_n^*, and E_n' families of equations provide us with additional tools with which to study equations holding in all QHLs but not all HLs. Importantly, they provide us with a property related to the field of the Hilbert space (and in particular holding in those Hilbert spaces with the classical fields of real numbers, complex numbers, and quaternions), something not previously known to be expressible by an equation.

Mayet showed that E_n follows from E_n^* (by setting $r = 0$), and he asked [8, p. 544] whether E_n and E_n^* are equivalent. The answer is affirmative.

Theorem 8.6. *In any* OML*, equation E_n is a consequence of E_n^* and vice-versa, for $n \geq 2$.*

Proof. We have already seen that E_n follows from E_n^*. For the converse, assume the hypotheses of E_n^* hold, i.e. that

$$((\Omega)\ \&\ r \perp a). \tag{8.5}$$

Starting from the l.h.s. of the E_n^* conclusion,

$$\begin{aligned}
&(a \cup r) \cap q \\
&\quad = (a_1 \cup \cdots \cup a_n \cup r) \cap (a_1 \cup b_1) \cap \cdots \cap (a_n \cup b_n) \\
&\quad \leq (a_1 \cup \cdots \cup a_n \cup r) \cap (a_1 \cup b_1 \cup r \cup r) \cap \cdots \cap (a_n \cup b_n \cup r \cup r) \\
&\quad = ((a_1 \cup \cdots \cup a_n) \cap (a_1 \cup b_1 \cup r) \cap \cdots \cap (a_n \cup b_n \cup r)) \cup r.
\end{aligned} \tag{8.6}$$

For the last step above, the distributive law is justified by F-H, since r commutes with each factor. Next, we substitute $b_1 \cup r$ for $b_1, \ldots, b_n \cup r$ for b_n into equation E_n. Since $a_i \perp b_i$ and $a_i \perp r$, we have $a_i \perp (b_i \cup r)$, so the hypotheses of this substitution instance of E_n are satisfied. The conclusion gives

$$(a_1 \cup \cdots \cup a_n) \cap (a_1 \cup (b_1 \cup r)) \cap \cdots \cap (a_n \cup (b_n \cup r)) \leq (b_1 \cup r) \cup \cdots \cup (b_n \cup r),$$

which, after rearrangements and joining r to the l.h.s.,

$$((a_1 \cup \cdots \cup a_n) \cap (a_1 \cup b_1 \cup r) \cap \cdots \cap (a_n \cup b_n \cup r)) \cup r \le b \cup r. \tag{8.7}$$

Chaining Eqs. (8.6) and (8.7), we conclude E_n^*. □

Mayet also asked whether an OML exists in which E_2^* fails. The answer is negative.

Corollary 8.7. *Equation* E_2^*,

$$\begin{aligned} &a_1 \perp b_1 \ \& \ a_2 \perp b_2 \ \& \ r \perp a_1 \ \& \ a_1 \perp a_2 \ \& \ a_2 \perp r \\ &\quad \Rightarrow (a_1 \cup a_2 \cup r) \cap (a_1 \cup b_1) \cap (a_2 \cup b_2) \le b_1 \cup b_2 \cup r, \end{aligned} \tag{8.8}$$

holds in all OML*s.*

Proof. Mayet showed [4] that E_2 holds in all OMLs, and the previous theorem shows that E_2 and E_2^* are equivalent in any OML. □

We can also generalise the previous Corollary, so as to replace the orthogonality relations with commutes relations (Definition 2.5) in all but the fourth hypotheses of Eq. (8.8).

Corollary 8.8. *The following generalisation of Equation* E_2^*,

$$\begin{aligned} &a_1 C b_1 \ \& \ a_2 C b_2 \ \& \ r C a_1 \ \& \ a_1 \perp a_2 \ \& \ a_2 C r \\ &\quad \Rightarrow (a_1 \cup a_2 \cup r) \cap (a_1 \cup b_1) \cap (a_2 \cup b_2) \le b_1 \cup b_2 \cup r \end{aligned} \tag{8.9}$$

holds in all OML*s.*

Proof. The proof runs as follows:

$$\begin{aligned} &(a_1 \cup a_2 \cup r) \cap (a_1 \cup b_1) \cap (a_2 \cup b_2) \\ &\quad = (a_1 \cup b_1) \cap ((a_2 \cup b_2) \cap (a_2 \cup (a_1 \cup r))) \\ &\quad = (a_1 \cup b_1) \cap (a_2 \cup (b_2 \cap (a_1 \cup r))) \\ &\quad \le (a_1 \cup b_1) \cap (a_1' \cup (b_2 \cap (a_1 \cup r))) \\ &\quad = (a_1 \cap (a_1' \cup (b_2 \cap (a_1 \cup r)))) \cup (b_1 \cap (a_1' \cup (b_2 \cap (a_1 \cup r)))) \\ &\quad \le (a_1 \cap (a_1' \cup (b_2 \cap (a_1 \cup r)))) \cup b_1 \\ &\quad \le ((a_1 \cup r) \cap (a_1' \cup (b_2 \cap (a_1 \cup r))) \cup b_1 \\ &\quad = ((a_1 \cup r) \cap a_1') \cup ((a_1 \cup r) \cap (b_2 \cap (a_1 \cup r))) \cup b_1 \\ &\quad \le ((a_1 \cup r) \cap a_1') \cup b_2 \cup b_1 \\ &\quad = ((a_1 \cap a_1') \cup (r \cap a_1')) \cup b_2 \cup b_1 \\ &\quad = 0 \cup (r \cap a_1') \cup b_2 \cup b_1 \\ &\quad \le b_1 \cup b_2 \cup r. \end{aligned}$$

In the second, fourth, seventh, and ninth steps we used F-H, applying the hypotheses of Eq. (8.9) as needed to obtain its prerequisite commuting conditions. In the third step, we used the hypothesis $a_1 \perp a_2$, i.e. $a_2 \le a_1'$. All other steps use simple ortholattice identities. □

9. Conclusion

In previous sections, we presented several results obtained in the field of Hilbert space equations, based on the states defined on the space. The idea is to use classes of Hilbert lattice equations for an alternative representation of Hilbert lattices and Hilbert spaces of arbitrary quantum systems that might eventually enable a direct introduction of the states of the systems into quantum computers. In applications, infinite classes could then be "truncated" to provide us with finite classes of required length. The obtained classes would in turn contribute to the theory of Hilbert space subspaces, which so far is poorly developed. And it is poorly developed because it turns out that to describe even the simplest physical system is a very demanding project.

In 1977, Hultgren and Shimony attempted to describe a spin-1 system by means of Greechie/Hasse diagrams/lattices using Stern–Gerlach devices [30]. Some of their diagrams were incomplete because, as shown Swift and Wright [31], they did not take into account both electric and magnetic fields. If they have done so, they could have patched the missing links in their Fig. 3 (dashed lines) and with them their lattice would read `123,456,789,ABC,58B`. However, even with that correction, their description cannot work because Greechie/Hasse diagrams are not subalgebras of a Hilbert lattice. Lattices necessary for a complete description of a quantum system[9] turn out to be too large for a brute-force approach in which we would first generate all possible lattices up to a very large number of atoms and then scan them to extract all required properties. Instead, we adopt a project of finding efficient algorithms that could enable us to carry out a partial descriptions of quantum systems in the lattice equation approach.

The algorithms and associated computer programs that were developed for this project were essential to its success. McKay's dynamic programming algorithm for n-Go equations (Sect. 6), together with its quickly convergent behaviour for large n, was particularly fortuitous. At the time of its discovery, no other way was known that could show the independence of the MGEs from all n-Go equations; at best, only empirical evidence pointing towards that answer could be accumulated. Indeed, this problem had remained open for nearly 20 years since Mayet's first publication [24] of these equations. Recently, Mayet found a direct proof of this independence that does not need a computer calculation [8]. Nonetheless, the

[9] Greechie diagrams describe orthogonalities between one-dimensional sublattices well, but, e.g. spans of nonorthogonal one-dimensional subspaces cannot be described by them at all—in the Hilbert space the corresponding subspaces are *not* equal to 1, i.e. they do not span the whole space while in a Greeche diagram they do. Consequently a proper lattice of a quantum system must be much larger than a Greechie diagram, which describes only orthogonalities between its spin components, because the Hilbert lattice equations require nonorthogonal atoms to pass a lattice. More specifically, nGO equations fail in Hasse/Greechie diagrams that describe only orthogonalities of Kochen–Specker setups but hold in their Hilbert space descriptions as well as in extended Hilbert lattices that take into account all needed relations between nonorthogonal atoms and their joins and meets [32].

algorithm still provides a useful tool for testing the simultaneous validity of all n-Go equations for individual OMLs that are not of the form required by Mayet's theorem.

Thus, there is a strong motivation to find other algorithms that, like McKay's n-Go dynamic programming algorithm, can be applied to other infinite families, in particular the nOA (generalised orthoarguesian) laws, Eq. (7.3). Assuming similar run-time behaviour could be achieved, these would provide us with extremely powerful tools that would let us test finite lattices against the family very quickly (instead of months or years of CPU time) as well as prove independence results for the entire infinite family at once (if the valuation set rapidly converges to a final, fixed value with increasing n, as it does for n-Go).

The success of the algorithm for n-Go depends crucially on the structure of a particular representation of the n-Go equations, where variables appear only on one side of the equation and are localised to an adjacent pair of conjuncts in a chain of conjuncts. Unfortunately, all currently known forms of the nOA laws have their variables distributed throughout their (very long) equations. So another approach, rather than finding a new algorithm, would be to discover a new form of the nOA laws that better separates their variable occurrences in such a way that the n-Go dynamic programming algorithm might be applicable. Both of these approaches are being investigated by the authors.

In Sect. 4, we described the application of LP to find states on a finite lattice. The authors are unaware of any previous use of LP methods for this purpose, in particular (for the present study) determining whether the lattice admits a strong set of states. There appear to be relatively few programs that deal with states, and most of the finite lattice examples in the literature related to states were found by hand. A Pascal program written by Klaey [33] is able to find certain kinds of states on lattices, but for the strong set of states problem it is apparently able to indicate only "yes" (if a strong set of states was found) or "unknown" otherwise. The LP method provides a definite answer either way, in the predictable amount of time that the simplex algorithm takes to run. Finally, the LP problem itself (with redundant constraints weakened) provides us with the information we need to construct a new Hilbert lattice equation that fails in a given lattice not admitting a strong set of states.

The `states.c` algorithm is actually more general than what we have described for the present work. It can also determine whether a finite lattice admits no states, exactly one state, a full set of states, a full set of dispersion-free ($\{0,1\}$) states, or group-valued states on the integers $\mathbb{Z}$ (Ref. [29] discusses lattices with some of these properties.)

Mayet's recent and important E-equation results [4] provide us with a powerful new method, the use of Hilbert space-valued states, to find previously unknown families of equations that hold in Hilbert lattices. For further investigation of these equations, it will be highly desirable to have a program analogous to our `states.c` (which works only with real-valued states) that will tell us whether or not a finite lattice admits a strong set of Hilbert space-valued states. This problem seems significantly harder than that of finding real-valued states, and possible algorithms for doing this are being explored by the authors.

In Sect. 6 (see the second half of the section), some new computational results on the Godowski lattices characterising Godowski equations are presented. In particular, we found out that the atom number increases for the successive smallest lattices in which Godowski equations of order n fail can be reduced from 6—as originally obtained by Godowski for any n—to 1 for $9 \leq n \leq 12$ (see Figs. 2 and 3) and most probably for all higher ns.

In Sect. 7, Theorem 7.3 tells us that further work is needed to determine whether or not Mayet's $\mathcal{E}_A$ equations are independent of the nOA laws. This problem is more difficult than it may first appear. With current techniques, all that we can do is either prove that a particular $\mathcal{E}_A$ equation can be derived from the nOA laws, or show that it is independent only up to some feasibly large n. Unlike the case with the n-Go laws, we have no known algorithm from showing independence from *all* equations in the infinite family of nOA laws. This open problem stresses the need to find such an algorithm.

In Sect. 8, Theorem 8.6 shows that two of Mayet's E-equation series, E_n, Eq. (8.2) and E^*_n, Eq. (8.3), are in fact equivalent (in an OML), answering an open question posed by Mayet [8, p. 544]. A third series, E'_n, Eq. (8.4), is strictly stronger than E_n (as already shown in Ref. [8, p. 549]).

The main results of Sect. 7 (Theorem 7.3) and Sect. 8 (Theorem 8.6) are both negative in the sense that they show that equations conjectured to be independent from others in fact are not. Nonetheless, such equations are still useful in the sense that they provide us with non-obvious new ways to express the equations they are equivalent to. In particular, they may move us a step closer to forms amenable to dynamic programming algorithms that would test entire infinite families at once.

The programs `latticego.c` and `states.c` described above can be downloaded from http://us.metamath.org/#ql.

Acknowledgements

Supported by the Ministry of Science, Education, and Sport of Croatia through the project no. 082-0982562-3160. Computational support was provided by the cluster Isabella of the University Computing Centre of the University of Zagreb and by the Croatian National Grid Infrastructure.

References

[1] Solèr, M.P.: Characterization of Hilbert spaces by orthomodular spaces. Comm. Alg. **23**, 219–243 (1995)

[2] Holland, S.S. Jr.: Orthomodularity in infinite dimensions; a theorem of M. Solèr. Bull. Am. Math. Soc. **32**, 205–234 (1995)

[3] Megill, N.D., Pavičić, M.: Equations, states, and lattices of infinite-dimensional Hilbert space. Int. J. Theor. Phys. **39**, 2337–2379 (2000)

[4] Mayet, R.: Equations holding in Hilbert lattices. Int. J. Theor. Phys. **45**, 1216–1246 (2006)

[5] Godowski, R.: Varieties of orthomodular lattices with a strongly full set of states. Demonstr. Math. **14**, 725–733 (1981)

[6] Mayet, R.: Equational bases for some varieties of orthomodular lattices related to states. Algebra Univers. **23**, 167–195 (1986)

[7] Pavičić, M., Megill, N.D.: Quantum logic and quantum computation. In: Engesser, K., Gabbay, D., Lehmann, D. (eds.) Handbook of Quantum Logic and Quantum Structures, vol. Quantum Structures, pp. 751–787. Elsevier, Amsterdam (2007)

[8] Mayet, R.: Ortholattice equations and Hilbert lattices. In: Engesser, K., Gabbay, D., Lehmann, D. (eds.) Handbook of Quantum Logic and Quantum Structures. Quantum Structures, pp. 525–554. Elsevier, Amsterdam (2007)

[9] Beran, L.: Orthomodular Lattices; Algebraic Approach. D. Reidel, Dordrecht (1985)

[10] Pavičić, M., Megill, N.D.: Is quantum logic a logic? In: Engesser, K., Gabbay, D., Lehmann, D. (eds.) Handbook of Quantum Logic and Quantum Structures. Quantum Logic, pp. 23–47. Elsevier, Amsterdam (2009)

[11] Birkhoff, G.: Lattice theory. In: American Mathematical Society Colloqium Publications, vol. XXV, 2nd (revised) edn. American Mathematical Society, New York (1948)

[12] Birkhoff, G.: Lattice theory. In: American Mathematical Society Colloquium Publications, vol. XXV, 3rd (new) edn. American Mathematical Society, Providence (1967)

[13] Pavičić, M.: Nonordered quantum logic and its YES–NO representation. Int. J. Theor. Phys. **32**, 1481–1505 (1993)

[14] Pavičić, M.: Identity rule for classical and quantum theories. Int. J. Theor. Phys. **37**, 2099–2103 (1998)

[15] Zeman, J.J.: Quantum logic with implications. Notre Dame J. Formal Logic **20**, 723–728 (1979)

[16] Beltrametti, E.G., Cassinelli, G.: The Logic of Quantum Mechanics. Addison-Wesley, Reading (1981)

[17] Kalmbach, G.: Measures and Hilbert Lattices. World Scientific, Singapore (1986)

[18] Ivert, P.-A., Sjödin, T.: On the impossibility of a finite propositional lattice for quantum mechanics. Helv. Phys. Acta **51**, 635–636 (1978)

[19] Jipsen, P., Rose, H.: Varieties of Lattices. Springer, New York (1992)

[20] Kalmbach, G.: Orthomodular Lattices. Academic Press, London (1983)

[21] Kalmbach, G.: Quantum Measures and Spaces. Kluwer, Dordrecht (1998)

[22] Mączyński, M.J.: Hilbert space formalism of quantum mechanics without the Hilbert space axiom. Rep. Math. Phys. **3**, 209–219 (1972)

[23] Pták, P., Pulmannová, S.: Orthomodular Structures as Quantum Logics. Kluwer, Dordrecht (1991)

[24] Mayet, R.: Varieties of orthomodular lattices related to states. Algebra Univers. **20**, 368–396 (1985)

[25] McKay, B.D., Megill, N.D., Pavičić, M.: Algorithms for Greechie diagrams. Int. J. Theor. Phys. **39**, 2381–2406 (2000)

[26] Pavičić, M., Merlet, J.-P., McKay, B.D., Megill, N.D.: Kochen–Specker vectors. J. Phys. A **38**, 1577–1592 and 3709 (corrigendum) (2005)

[27] Jajte, R., Paszkiewicz, A.: Vector measures on the closed subspaces of a Hilbert space. Studia Math. **63**, 229–251 (1978)

[28] Maeda, F., Maeda, S.: Theory of Symmetric Lattices. Springer, New York (1970)

[29] Navara, M.: Small quantum structures with small state spaces. Int. J. Theor. Phys. **47**, 36–43 (2008)

[30] Hultgren, B.O. III., Shimony, A.: The lattice of verifiable propositions of the spin-1 system. J. Math. Phys. **18**, 381–394 (1977)

[31] Swift, A.R., Wright, R.: Generalized Stern–Gerlach experiments and the observability of arbitrary spin operators. J. Math. Phys. **21**, 77–82 (1980)

[32] Pavičić, M., McKay, B.D., Megill, N.D., Fresl, K.: Reverse Kochen–Specker setups (2009, in preparation)

[33] Kläy, M.: Stochastic Models on Empirical Systems, Empirical Logic and Quantum Logics, and States on Hypergraphs. PhD thesis, University of Bern, Faculty of Natural Science, Fischer Druck, Münsingen (1985)

Norman D. Megill
Boston Information Group
19 Locke Ln.
Lexington, MA 02420, USA
e-mail: nm@alum.mit.edu

Mladen Pavičić
Physics Chair
Faculty of Civil Engineering
University of Zagreb
Zagreb, Croatia
e-mail: pavicic@grad.hr

Communicated by Carlo Rovelli.

Received: June 25, 2009.

Accepted: December 8, 2009.