# Shall we find normal forms in orthomodular lattices?

Jeannine Gabriëls

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic

FSTA 2012, Liptovský Ján

# Outline

- Definitions

# Outline

- Definitions
- Motivation of today's talk

# Outline

- Definitions
- Motivation of today's talk
- Tools

# Outline

- Definitions
- Motivation of today's talk
- Tools
- Results

# Outline

- Definitions
- Motivation of today's talk
- Tools
- Results
- Conclusion

# Definitions

## ortholattice, modular lattice, modular ortho- and orthomodular lattice

- an *ortholattice* is a lattice with an ortho-complementation

| | | | |
|---|---|---|---|
| order-reversing | $a \leq b$ | $\Rightarrow$ | $b' \leq a'$ |
| involution law | $a''$ | $=$ | $a$ |
| complement law | $a' \vee a$ | $=$ | $1$ |
| | $a' \wedge a$ | $=$ | $0$ |

# Definitions

## ortholattice, modular lattice, modular ortho- and orthomodular lattice

- an *ortholattice* is a lattice with an ortho-complementation

$$\begin{array}{lrcl} \text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\ \text{involution law} & a'' & = & a \\ \text{complement law} & a' \vee a & = & 1 \\ & a' \wedge a & = & 0 \end{array}$$

- a *modular lattice* is a lattice in which the modular law holds

$$a \vee (b \wedge (a \vee c)) = (a \vee b) \wedge (a \vee c)$$

# Definitions

## orthollattice, modular lattice, modular ortho- and orthomodular lattice

- an *ortholattice* is a lattice with an orthocomplementation

$$\begin{array}{llcl} \text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\ \text{involution law} & a'' & = & a \\ \text{complement law} & a' \vee a & = & 1 \\ & a' \wedge a & = & 0 \end{array}$$

- a *modular lattice* is a lattice in which the modular law holds

$$a \vee (b \wedge (a \vee c)) = (a \vee b) \wedge (a \vee c)$$

- a *modular ortholattice* is a lattice which is orthocomplemented and modular

# Definitions

## ortholattice, modular lattice, modular ortho- and orthomodular lattice

- an *ortholattice* is a lattice with an ortho-complementation

$$
\begin{array}{llcl}
\text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\
\text{involution law} & a'' & = & a \\
\text{complement law} & a' \vee a & = & 1 \\
 & a' \wedge a & = & 0
\end{array}
$$

- a *modular lattice* is a lattice in which the modular law holds

$$a \vee (b \wedge (a \vee c)) = (a \vee b) \wedge (a \vee c)$$

- a *modular ortholattice* is a lattice which is orthocomplemented and modular
- an *orthomodular lattice* is a ortholattice in which the orthomodular law holds

# Definitions

## word

- In an algebraic sense a *word* is a formal expression or finite string of symbols build up in variables and algebraic operations

# Definitions

## word

- In an algebraic sense a *word* is a formal expression or finite string of symbols build up in variables and algebraic operations

  Each word represents a particular element of the algebra, which is generated by the given variables and which is closed with respect to the given operations

# Definitions

## word problem, decidability, normal form

- A *word problem* is the problem of deciding, whether or not two given words represent the same element of the algebra.

# Definitions

## word problem, decidability, normal form

- A *word problem* is the problem of deciding, whether or not two given words represent the same element of the algebra.
- If there exists such an algorithm, then we say the word problem is *decidable* (*solvable*), otherwise it is *undecidable* (*unsolvable*).

# Definitions

## word problem, decidability, normal form

- A *word problem* is the problem of deciding, whether or not two given words represent the same element of the algebra.
- If there exists such an algorithm, then we say the word problem is *decidable* (*solvable*), otherwise it is *undecidable* (*unsolvable*).
- A *normal form* also called *canonical form* of an object is a standard way of presenting that object.

# Definitions

## word problem, decidability, normal form

- A *word problem* is the problem of deciding, whether or not two given words represent the same element of the algebra.
- If there exists such an algorithm, then we say the word problem is *decidable* (*solvable*), otherwise it is *undecidable* (*unsolvable*).
- A *normal form* also called *canonical form* of an object is a standard way of presenting that object.

  Philip M. Whitman proved that for all equal elements in a free lattice, there is one of shortest length.

Max W. Dehn knew that the word problem was difficult, he wrote:

> *Solving the word problem for all groups may be as impossible as solving all mathematical problems.*

Max W. Dehn knew that the word problem was difficult, he wrote:

> *Solving the word problem for all groups may be as impossible as solving all mathematical problems.*

P. Novikov (1955) and independently, W. Boone (1958) proved that the word problem is in general not solvable.

Max W. Dehn knew that the word problem was difficult, he wrote:

> *Solving the word problem for all groups may be as impossible as solving all mathematical problems.*

P. Novikov (1955) and independently, W. Boone (1958) proved that the word problem is in general not solvable.

Skolem (1920) solved the (uniform) word problem for finitely presented lattices.

# What about quantum logic?

Is the word problem related to quantum logic decidable?

# What about quantum logic?

Is the word problem related to quantum logic decidable?

First results:

# What about quantum logic?

Is the word problem related to quantum logic decidable?

First results:

- for free ortholattices the word problem is decidable (G. Bruns, A. Meinander)

# What about quantum logic?

Is the word problem related to quantum logic decidable?

First results:

- for free ortholattices the word problem is decidable (G. Bruns, A. Meinander)
- but for free modular lattices $M(n)$ it is undecidable when $n \geq 4$

# What about quantum logic?

Is the word problem related to quantum logic decidable?

First results:

- for free ortholattices the word problem is decidable (G. Bruns, A. Meinander)
- but for free modular lattices $M(n)$ it is undecidable when $n \geq 4$
- the word problem remains an open challenge in the orthomodular as well as in the modular-ortho case (Herrmann, Micol and Roddy)

# What about quantum logic?

Is the word problem related to quantum logic decidable?

First results:

- for free ortholattices the word problem is decidable (G. Bruns, A. Meinander)
- but for free modular lattices $M(n)$ it is undecidable when $n \geq 4$
- the word problem remains an open challenge in the orthomodular as well as in the modular-ortho case (Herrmann, Micol and Roddy)
- for the free orthomodular lattices over two generators, the problem is decidable

# Motivation of today's talk

The absence of distributivity in OMLs makes it difficult to find the normal forms of complex expressions. Some tools were developed to overcome this problem:

# Motivation of today's talk

The absence of distributivity in OMLs makes it difficult to find the normal forms of complex expressions. Some tools were developed to overcome this problem:

Foulis-Holland Theorem

# Motivation of today's talk

The absence of distributivity in OMLs makes it difficult to find the normal forms of complex expressions. Some tools were developed to overcome this problem:

Foulis-Holland Theorem

Focusing technique (Greechie)

# Motivation of today's talk

The absence of distributivity in OMLs makes it difficult to find the normal forms of complex expressions. Some tools were developed to overcome this problem:

Foulis-Holland Theorem

Focusing technique (Greechie)

Computation in $F(a, b)$ (Navara)

# Motivation of today's talk

The absence of distributivity in OMLs makes it difficult to find the normal forms of complex expressions. Some tools were developed to overcome this problem:

Foulis-Holland Theorem

Focusing technique (Greechie)

Computation in $F(a, b)$ (Navara)

In all three techniques the commuting elements play a crucial role

$$xCy \quad \text{if} \quad x = (x \wedge y) \vee (x \wedge y')$$

# Today's talk

For which of the 96 binary operations $*$ in an OML, the following implication holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML.

# Why this question?

Is it possible to find operations $*$ for which

$$(a_1 * b) \wedge (a_2 * b) \wedge \ldots \wedge (a_n * b) = (a_1 \wedge \ldots \wedge a_n) * b$$

holds?

# Tools

- There are $2^4 = 16$ binary Boolean operations, each of them represents 6 OML operations (Megill, Pavičić)

# Tools

- There are $2^4 = 16$ binary Boolean operations, each of them represents 6 OML operations (Megill, Pavičić)
- Navara's computation in $F(a, b, c)$

# Tools

- There are $2^4 = 16$ binary Boolean operations, each of them represents 6 OML operations (Megill, Pavičić)
- Navara's computation in $F(a, b, c)$
- Kalmbach embedding

# Tools

- There are $2^4 = 16$ binary Boolean operations, each of them represents 6 OML operations (Megill, Pavičić)
- Navara's computation in $F(a, b, c)$
- Kalmbach embedding
- Computer program (Hyčko)
  http://www.mat.savba.sk/~hycko/oml

# Boolean algebra

For which of the 16 binary operations $*$ in an Boolean algebra, the following implication holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

for $x$, $y$ and $z$, elements of an Boolean algebra.

# Boolean algebra

From the Boolean operations only two do not fulfil the equation

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

# Boolean algebra

From the Boolean operations only two do not fulfil the equation

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

the equivalence $\leftrightarrow$

$$a \leftrightarrow b \;=\; (a \wedge b) \vee (a' \wedge b') \;=\; \begin{cases} 1 & \text{if } a = b\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

# Boolean algebra

From the Boolean operations only two do not fulfil the equation

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

the equivalence $\leftrightarrow$

$$a \leftrightarrow b \;=\; (a \wedge b) \vee (a' \wedge b') \;=\; \begin{cases} 1 & \text{if } a = b\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

and its complement $\not\leftrightarrow$

$$a \not\leftrightarrow b \;=\; (a \wedge b') \vee (a' \wedge b) \;=\; \begin{cases} 0 & \text{if } a = b\,, \\ 1 & \text{otherwise}\,. \end{cases}$$

# Corresponding in an OML

These two operations correspond to 12 OML operations

# Corresponding in an OML

These two operations correspond to 12 OML operations

- For the equivalence $\leftrightarrow$

| | | |
|---|---|---|
| $(x \wedge y) \vee (x' \wedge y')$ | Beran 8 |  |
| $(x' \vee y) \wedge [x \vee (x' \wedge y')]$ | Beran 24 |  |
| $(x \vee y') \wedge [y \vee (x' \wedge y')]$ | Beran 40 |  |
| $(x' \vee y) \wedge [y' \vee (x \wedge y)]$ | Beran 56 |  |
| $(x \vee y') \wedge [x' \vee (x \wedge y)]$ | Beran 72 |  |
| $(x' \vee y) \wedge (x \vee y')$ | Beran 88 |  |

# Corresponding in an OML

- For its complement $\not\leftrightarrow$

| | | |
|---|---|---|
| $(x \wedge y') \vee (x' \wedge y)$ | Beran 9 |  |
| $(x' \vee y') \wedge [x \vee (x' \wedge y)]$ | Beran 25 |  |
| $(x' \vee y') \wedge [y \vee (x \wedge y')]$ | Beran 41 |  |
| $(x \vee y) \wedge [y' \vee (x' \wedge y)]$ | Beran 57 |  |
| $(x \vee y) \wedge [x' \vee (x \wedge y')]$ | Beran 73 |  |
| $(x \vee y) \wedge (x' \vee y')$ | Beran 89 |  |

# Calculation in $F(a, b, c)$

For two elements $x$, $y$ of an OML, it is said $x$ commutes with $y$ and written $xCy$ if

$$x = (x \wedge y) \vee (x \wedge y'),$$

# Calculation in $F(a, b, c)$

For two elements $x$, $y$ of an OML, it is said $x$ commutes with $y$ and written $xCy$ if

$$x = (x \wedge y) \vee (x \wedge y'),$$

It is easy to show that the following holds:

$$x \leq y \;\Rightarrow\; xCy.$$

# Calculation in $F(a, b, c)$

For two elements $x$, $y$ of an OML, it is said $x$ commutes with $y$ and written $xCy$ if

$$x = (x \wedge y) \vee (x \wedge y'),$$

It is easy to show that the following holds:

$$x \leq y \ \Rightarrow \ xCy.$$

M. Navara described the OML $F(a, b, c)$ generated by three generators $a, b, c$ where $cCa$ and $cCb$.

# Calculation in $F(a, b, c)$

The generators $a$, $b$ and $c$ can be expressed by Navara's notation:

$$
\begin{aligned}
a &= a\,(\;,\;)_c\, b \\
b &= a\,(\;,\;)_c\, b \\
c &= a\,(\;,\;)_c\, b\,.
\end{aligned}
$$

# Calculation in $F(a, b, c)$

The generators $a$, $b$ and $c$ can be expressed by Navara's notation:

$a = a($  ,  $)_c\, b$

$b = a($  ,  $)_c\, b$

$c = a($  ,  $)_c\, b\,.$

$$a \wedge c \in [0, c]$$

$$b \wedge c' \in [0, c']$$

# Calculation in $F(a, b, c)$

Our initial question:

# Calculation in $F(a, b, c)$

Our initial question:
For which binary operations $*$ holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML,

# Calculation in $F(a, b, c)$

Our initial question:
For which binary operations $*$ holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML,
becomes by substituting

# Calculation in $F(a, b, c)$

Our initial question:
For which binary operations $*$ holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML,
becomes by substituting $x$ by $(a \wedge c)$,

# Calculation in $F(a, b, c)$

Our initial question:
For which binary operations $*$ holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML,
becomes by substituting $x$ by $(a \wedge c)$, $y$ by $c$

# Calculation in $F(a, b, c)$

Our initial question:
For which binary operations $*$ holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML,
becomes by substituting $x$ by $(a \wedge c)$, $y$ by $c$ and $z$ by $b$.

# Calculation in $F(a, b, c)$

Our initial question:
For which binary operations $*$ holds:

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

where $x$, $y$ and $z$ are elements of an OML,
becomes by substituting $x$ by $(a \wedge c)$, $y$ by $c$ and $z$ by $b$.
For which binary operations $*$ holds:

$$(a \wedge c) * b \leq c * b$$

where $a$, $b$ and $c$ are the generators of $F(a, b, c)$.

# Calculation in $F(a, b, c)$

- By choosing $c = 1$:

# Calculation in $F(a, b, c)$

- By choosing $c = 1$:

$$a * b \leq 1 * b$$

# Calculation in $F(a, b, c)$

- By choosing $c = 1$:

  $$a * b \leq 1 * b$$

We could eliminate further 40 operations from our list of candidates.

# Calculation in $F(a, b, c)$

- By choosing $c = 1$:

$$a * b \leq 1 * b$$

We could eliminate further 40 operations from our list of candidates.

Do the remaining 44 binary operations fulfil our equation?

# Kalmbach embedding

- A method of embedding any arbitrary poset $P$ into a concrete OML $L = K(L)$, used by Kalmbach and extended by Harding (1991) and Mayet and Navara (1995).

# Kalmbach embedding

- A method of embedding any arbitrary poset $P$ into a concrete OML $L = K(L)$, used by Kalmbach and extended by Harding (1991) and Mayet and Navara (1995).
- Given any poset $P$, then there exists an OML $L$ and an embedding
  $\phi : P \to L$

# Kalmbach embedding

- A method of embedding any arbitrary poset $P$ into a concrete OML $L = K(L)$, used by Kalmbach and extended by Harding (1991) and Mayet and Navara (1995).
- Given any poset $P$, then there exists an OML $L$ and an embedding
  $\phi : P \to L$
  for $x, y \in P$
  1. $x \leq y \Leftrightarrow \phi(x) \leq \phi(y)$
  2. if $x \wedge y$ exists, then $\phi(x) \wedge \phi(y) = \phi(x \wedge y)$
  3. if $x \vee y$ exists, then $\phi(x) \vee \phi(y) = \phi(x \vee y)$

# Kalmbach embedding

- A method of embedding any arbitrary poset $P$ into a concrete OML $L = K(L)$, used by Kalmbach and extended by Harding (1991) and Mayet and Navara (1995).
- Given any poset $P$, then there exists an OML $L$ and an embedding
  $\phi : P \to L$
  for $x, y \in P$
  1. $x \leq y \Leftrightarrow \phi(x) \leq \phi(y)$
  2. if $x \wedge y$ exists, then $\phi(x) \wedge \phi(y) = \phi(x \wedge y)$
  3. if $x \vee y$ exists, then $\phi(x) \vee \phi(y) = \phi(x \vee y)$
- $L$ can then be embedded into a Boolean algebra by preserving the lattice operations.

# Kalmbach embedding

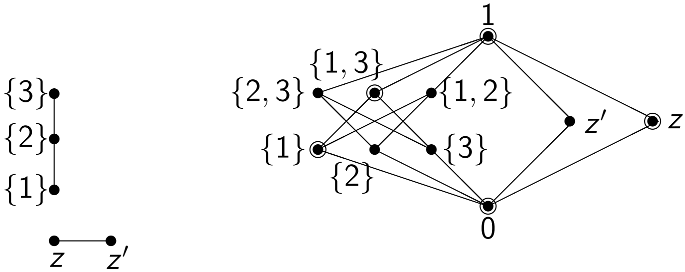



Figure: Greechie- and Hasse diagram of $2^3 \oplus 2^2$

# Kalmbach embedding

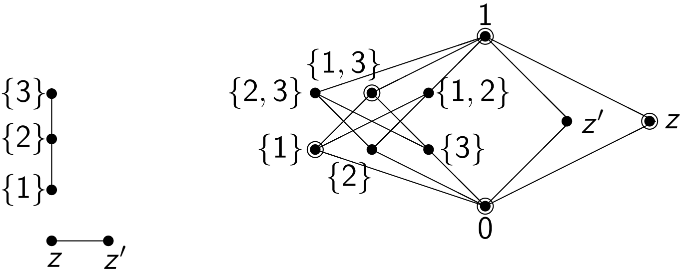
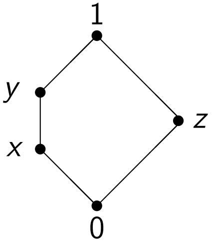



Figure: Greechie- and Hasse diagram of $2^3 \oplus 2^2$

Which is the Kalmbach embedding of the pentagon.

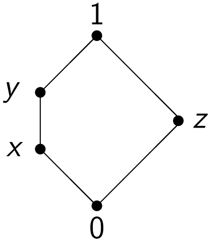
# Kalmbach embedding

## Monotony in the first argument



# Kalmbach embedding

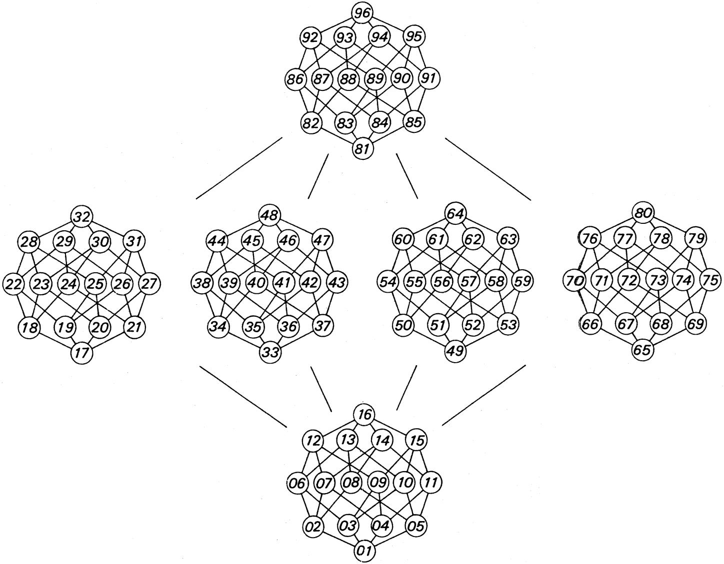
## Monotony in the first argument




This demonstrates that, for exploring the monotonicity in the first argument, we can discard all the operations $*$ with Beran's number in $\{65, \ldots, 80\}$.

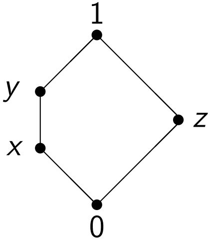
# Kalmbach embedding

## Monotony in the first argument



# Kalmbach embedding

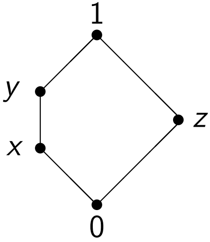
## Monotony in the first argument




This demonstrates that, for exploring the monotonicity in the first argument, we can discard all the operations $*$ with Beran's number in $\{65, \ldots, 80\}$.

# Kalmbach embedding

## Monotony in the first argument




This demonstrates that, for exploring the monotonicity in the first argument, we can discard all the operations $*$ with Beran's number in $\{65, \ldots, 80\}$.
Further, we can even discard all the binary operations containing $a'$.

# Results

## Monotony in the first argument

At the end only 17 binary operations come into question to be non-decreasing in the first argument, they are the operations with Beran's number and Navara's notation:

# Results

## Monotony in the first argument

At the end only 17 binary operations come into question to be non-decreasing in the first argument, they are the operations with Beran's number and Navara's notation:

1 , 2 , 3 , 6 ,

# Results

## Monotony in the first argument

At the end only 17 binary operations come into question to be non-decreasing in the first argument, they are the operations with Beran's number and Navara's notation:

1 , 2 , 3 , 6 ,

22, 

# Results

## Monotony in the first argument

At the end only 17 binary operations come into question to be non-decreasing in the first argument, they are the operations with Beran's number and Navara's notation:

1 , 2 , 3 , 6 ,

22, 

34 , 38 , 39 , 44 ,

# Results

## Monotony in the first argument

At the end only 17 binary operations come into question to be non-decreasing in the first argument, they are the operations with Beran's number and Navara's notation:

1 , 2 , 3 , 6 ,

22, 

34 , 38 , 39 , 44 ,

51 , 54 , 58 , 61 ,

# Results

## Monotony in the first argument

At the end only 17 binary operations come into question to be non-decreasing in the first argument, they are the operations with Beran's number and Navara's notation:

1 , 2 , 3 , 6 ,

22, 

34 , 38 , 39 , 44 ,

51 , 54 , 58 , 61 ,

86 , 92 , 93  and 96 .

# Results

These are exactly the binary operations for which holds

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

# Results

These are exactly the binary operations for which holds

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

We found also 17 operations which are non-decreasing in the second argument,

# Results

6 operations are non-decreasing in both arguments,

# Results

6 operations are non-decreasing in both arguments,

0 Beran's number 1 

# Results

6 operations are non-decreasing in both arguments,

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a \wedge b$ | Beran's number 2 |  |

# Results

6 operations are non-decreasing in both arguments,

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a \wedge b$ | Beran's number 2 |  |
| $a$ | Beran's number 22 |  |

# Results

6 operations are non-decreasing in both arguments,

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a \wedge b$ | Beran's number 2 |  |
| $a$ | Beran's number 22 |  |
| $b$ | Beran's number 39 |  |

# Results

6 operations are non-decreasing in both arguments,

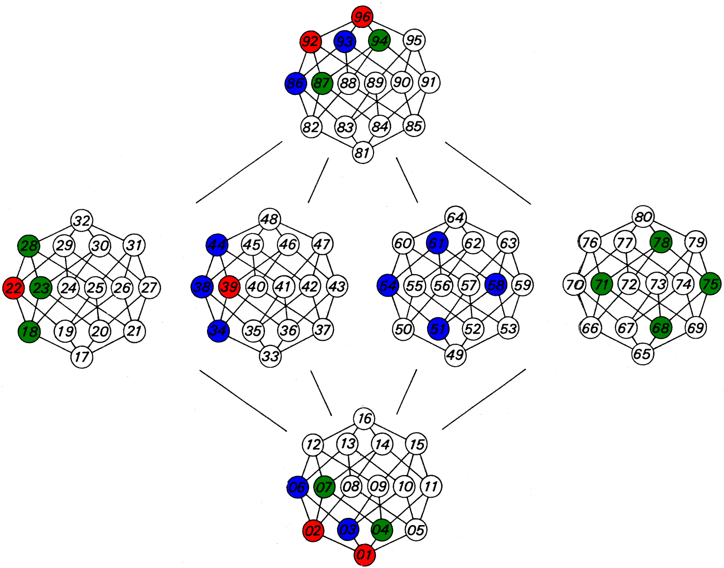
| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a \wedge b$ | Beran's number 2 |  |
| $a$ | Beran's number 22 |  |
| $b$ | Beran's number 39 |  |
| $a \vee b$ | Beran's number 92 |  |

# Results

6 operations are non-decreasing in both arguments,

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a \wedge b$ | Beran's number 2 |  |
| $a$ | Beran's number 22 |  |
| $b$ | Beran's number 39 |  |
| $a \vee b$ | Beran's number 92 |  |
| $1$ | Beran's number 96 |  |

# Results


96
92
93
94
95
86
87
88
89
90
91
82
83
84
85
81
32
28
29
30
31
22
23
24
25
26
27
18
19
20
21
17
48
44
45
46
47
38
39
40
41
42
43
34
35
36
37
33
64
60
61
62
63
54
55
56
57
58
59
50
51
52
53
49
80
76
77
78
79
70
71
72
73
74
75
66
67
68
69
65
16
12
13
14
15
06
07
08
09
10
11
02
03
04
05
01

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

0 Beran's number 1

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a' \wedge b'$ | Beran's number 5 |  |

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a' \wedge b'$ | Beran's number 5 |  |
| $a'$ | Beran's number 75 |  |

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a' \wedge b'$ | Beran's number 5 |  |
| $a'$ | Beran's number 75 |  |
| $b'$ | Beran's number 58 |  |

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a' \wedge b'$ | Beran's number 5 |  |
| $a'$ | Beran's number 75 |  |
| $b'$ | Beran's number 58 |  |
| $a' \vee b'$ | Beran's number 95 |  |

# Results

We also reversed our initial inequation For which of the 96 binary operations $*$ in an OML, the following inequality holds:

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

| | | |
|---|---|---|
| $0$ | Beran's number 1 |  |
| $a' \wedge b'$ | Beran's number 5 |  |
| $a'$ | Beran's number 75 |  |
| $b'$ | Beran's number 58 |  |
| $a' \vee b'$ | Beran's number 95 |  |
| $1$ | Beran's number 96 |  |

# Results

After all there are 46 binary operations which fulfil

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

or

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

in the first or second argument

# Results

After all there are 46 binary operations which fulfil

$$x \leq y \quad \Rightarrow \quad x * z \leq y * z$$

or

$$x \leq y \quad \Rightarrow \quad x * z \geq y * z$$

in the first or second argument
and if $a * b$ is one of them then also

$$a' * b$$
$$b * a$$
$$b' * a$$

# Conclusions

Our aim was to reduce the complexity of some OML operations and to find a way to write them in a unique normal form.

# Conclusions

Our aim was to reduce the complexity of some OML operations and to find a way to write them in a unique normal form.
Therefor we need to find two operations which satisfy the distributive law.

# Conclusions

Our aim was to reduce the complexity of some OML operations and to find a way to write them in a unique normal form.
Therefor we need to find two operations which satisfy the distributive law.
One of these operations need also to be associative and commutative.

# Conclusions

Our aim was to reduce the complexity of some OML operations and to find a way to write them in a unique normal form.
Therefor we need to find two operations which satisfy the distributive law.
One of these operations need also to be associative and commutative.
The only two operations which fulfil the three conditions are the join and the meet.

# Conclusions

Our aim was to reduce the complexity of some OML operations and to find a way to write them in a unique normal form.
Therefor we need to find two operations which satisfy the distributive law.
One of these operations need also to be associative and commutative.
The only two operations which fulfil the three conditions are the join and the meet.
so, we had to find the operations which distribute over the meet

$$(a_1 * b) \wedge (a_2 * b) \wedge \ldots \wedge (a_n * b) = (a_1 \wedge \ldots \wedge a_n) * b$$

# Conclusions

# Conclusions

- Standard methods of Boolean algebras are not applicable in OMLs.

# Conclusions

- Standard methods of Boolean algebras are not applicable in OMLs.

- Shall we find normal forms in orthomodular lattices?

- Thank you for your attention!

- Thank you for your attention!
- Questions?

# Definitions

- an *ortholattice* is a lattice with an orthocomplementation

$$\begin{array}{lrcl} \text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\ \text{involution law} & a'' & = & a \\ \text{complement law} & a' \vee a & = & 1 \\ & a' \wedge a & = & 0 \end{array}$$

# Definitions

- an *ortholattice* is a lattice with an ortho-complementation

$$\begin{array}{lrcl} \text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\ \text{involution law} & a'' & = & a \\ \text{complement law} & a' \vee a & = & 1 \\ & a' \wedge a & = & 0 \end{array}$$

- a *modular lattice* is a lattice in which the modular law holds

$$a \vee (b \wedge (a \vee c)) = (a \vee b) \wedge (a \vee c)$$

# Definitions

- an *ortholattice* is a lattice with an ortho-complementation

$$
\begin{array}{lrcl}
\text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\
\text{involution law} & a'' & = & a \\
\text{complement law} & a' \vee a & = & 1 \\
 & a' \wedge a & = & 0
\end{array}
$$

- a *modular lattice* is a lattice in which the modular law holds

$$a \vee (b \wedge (a \vee c)) = (a \vee b) \wedge (a \vee c)$$

- a *modular ortholattice* is a lattice which is orthocomplemented and modular

# Definitions

- an *ortholattice* is a lattice with an ortho-complementation

$$\begin{array}{lrcl} \text{order-reversing} & a \leq b & \Rightarrow & b' \leq a' \\ \text{involution law} & a'' & = & a \\ \text{complement law} & a' \vee a & = & 1 \\ & a' \wedge a & = & 0 \end{array}$$

- a *modular lattice* is a lattice in which the modular law holds

$$a \vee (b \wedge (a \vee c)) = (a \vee b) \wedge (a \vee c)$$

- a *modular ortholattice* is a lattice which is orthocomplemented and modular
- an *orthomodular lattice* is a ortholattice in which the orthomodular law holds

$$a \leq b \Rightarrow b = a \vee (a' \wedge b)$$