Some results obtained with the Universal Algebra Calculator, theorem provers and constraint satisfiers in a Sage package

Peter Jipsen

Chapman University, Orange, California

March 19, 2011

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

Sage (sagemath.org) is a free open-source computer algebra system

It combines many research tools with a convenient browser interface

E.g. includes GAP (for Groups, Algorithms and Programs)

Maxima for symbolic algebra

Networkx for graph theory

R for statistics, etc ...

Python is used to interact with and program in Sage

But Sage does not have direct support for Universal Algebra

The UA Calculator (uacalc.org) by E. Kiss, R. Freese and M. Valeriote contains highly optimized algorithms for computations in a finite algebra

Can calculate Con(A), Sub(A), A^n, $F_{V(A)}(n)$, ...

Has a Java graphical interface for displaying (congruence) lattices

But Sage does not have direct support for Universal Algebra

The UA Calculator (uacalc.org) by E. Kiss, R. Freese and M. Valeriote contains highly optimized algorithms for computations in a finite algebra

Can calculate Con(A), Sub(A), A^n, $F_{V(A)}(n)$, ...

Has a Java graphical interface for displaying (congruence) lattices

But Sage does not have direct support for Universal Algebra

The UA Calculator (uacalc.org) by E. Kiss, R. Freese and M. Valeriote contains highly optimized algorithms for computations in a finite algebra

Can calculate Con(A), Sub(A), A^n, $F_{V(A)}(n)$, ...

Has a Java graphical interface for displaying (congruence) lattices

But Sage does not have direct support for Universal Algebra

The UA Calculator (uacalc.org) by E. Kiss, R. Freese and M. Valeriote contains highly optimized algorithms for computations in a finite algebra

Can calculate Con(A), Sub(A), A^n, $F_{V(A)}(n)$, ...

Has a Java graphical interface for displaying (congruence) lattices

But algebras have to be entered or loaded one at a time

Computed results are not easy to export

Difficult to program additional algorithms or extend interface

Instead wrote some small Java routines that allow Sage to access the UA Calculator

Sage can handle lists of algebras, e.g. compute Con(A) for each of them

But algebras have to be entered or loaded one at a time

Computed results are not easy to export

Difficult to program additional algorithms or extend interface

Instead wrote some small Java routines that allow Sage to access the UA Calculator

Sage can handle lists of algebras, e.g. compute Con(A) for each of them

But algebras have to be entered or loaded one at a time

Computed results are not easy to export

Difficult to program additional algorithms or extend interface

Instead wrote some small Java routines that allow Sage to access the UA Calculator

Sage can handle lists of algebras, e.g. compute $\mathrm{Con}(A)$ for each of them

But algebras have to be entered or loaded one at a time

Computed results are not easy to export

Difficult to program additional algorithms or extend interface

Instead wrote some small Java routines that allow Sage to access the UA Calculator

Sage can handle lists of algebras, e.g. compute Con(A) for each of them

But algebras have to be entered or loaded one at a time

Computed results are not easy to export

Difficult to program additional algorithms or extend interface

Instead wrote some small Java routines that allow Sage to access the UA Calculator

Sage can handle lists of algebras, e.g. compute Con(A) for each of them

Implemented Heitzig and Reinhold's method for enumerating finite lattices in Python

Computed all lattices with up to 12 elements

Filtered the list to get only subdirectly irreducible lattices

Task: compute (the bottom of) the poset of join-irreducible subvarieties $V(A)$ of Λ_{Lat}

Since Λ_{Lat} is distributive, the downsets of this poset form Λ_{Lat} (f.g. part)

Implemented Heitzig and Reinhold's method for enumerating finite lattices in Python

Computed all lattices with up to 12 elements

Filtered the list to get only subdirectly irreducible lattices

Task: compute (the bottom of) the poset of join-irreducible subvarieties $V(A)$ of Λ_{Lat}

Since Λ_{Lat} is distributive, the downsets of this poset form Λ_{Lat} (f.g. part)

Implemented Heitzig and Reinhold's method for enumerating finite lattices in Python

Computed all lattices with up to 12 elements

Filtered the list to get only subdirectly irreducible lattices

Task: compute (the bottom of) the poset of join-irreducible subvarieties $V(A)$ of Λ_{Lat}

Since Λ_{Lat} is distributive, the downsets of this poset form Λ_{Lat} (f.g. part)

Implemented Heitzig and Reinhold's method for enumerating finite lattices in Python

Computed all lattices with up to 12 elements

Filtered the list to get only subdirectly irreducible lattices

Task: compute (the bottom of) the poset of join-irreducible subvarieties $V(A)$ of Λ_{Lat}

Since Λ_{Lat} is distributive, the downsets of this poset form Λ_{Lat} (f.g. part)

Implemented Heitzig and Reinhold's method for enumerating finite lattices in Python

Computed all lattices with up to 12 elements

Filtered the list to get only subdirectly irreducible lattices

Task: compute (the bottom of) the poset of join-irreducible subvarieties $V(A)$ of Λ_{Lat}

Since Λ_{Lat} is distributive, the downsets of this poset form Λ_{Lat} (f.g. part)

For finite s.i. lattices A, B we have $V(A) \subseteq V(B)$ iff $A \in HS(B)$

So how does one compute $HS(B)$?

Use the UA Calculator to find Sub(B)

Eliminate isomorphic copies to get $S(B)$

For each $C \in S(B)$ test if $A \in H(C)$, but how?

Checking all maps from C to A is not feasible

For finite s.i. lattices A, B we have $V(A) \subseteq V(B)$ iff $A \in HS(B)$

So how does one compute $HS(B)$?

Use the UA Calculator to find $\mathrm{Sub}(B)$

Eliminate isomorphic copies to get $S(B)$

For each $C \in S(B)$ test if $A \in H(C)$, but how?

Checking all maps from C to A is not feasible

For finite s.i. lattices A, B we have $V(A) \subseteq V(B)$ iff $A \in HS(B)$

So how does one compute $HS(B)$?

Use the UA Calculator to find $\mathrm{Sub}(B)$

Eliminate isomorphic copies to get $S(B)$

For each $C \in S(B)$ test if $A \in H(C)$, but how?

Checking all maps from C to A is not feasible

For finite s.i. lattices A, B we have $V(A) \subseteq V(B)$ iff $A \in HS(B)$

So how does one compute $HS(B)$?

Use the UA Calculator to find $\mathrm{Sub}(B)$

Eliminate isomorphic copies to get $S(B)$

For each $C \in S(B)$ test if $A \in H(C)$, but how?

Checking all maps from C to A is not feasible

For finite s.i. lattices A, B we have $V(A) \subseteq V(B)$ iff $A \in HS(B)$

So how does one compute $HS(B)$?

Use the UA Calculator to find $\mathrm{Sub}(B)$

Eliminate isomorphic copies to get $S(B)$

For each $C \in S(B)$ test if $A \in H(C)$, but how?

Checking all maps from C to A is not feasible

For finite s.i. lattices A, B we have $V(A) \subseteq V(B)$ iff $A \in HS(B)$

So how does one compute $HS(B)$?

Use the UA Calculator to find $\mathrm{Sub}(B)$

Eliminate isomorphic copies to get $S(B)$

For each $C \in S(B)$ test if $A \in H(C)$, but how?

Checking all maps from C to A is not feasible

Instead use a Constraint Satisfaction solver, e.g. Minion (minion.sourceforge.net)

A *constraint satisfaction problem* (CSP) is traditionally given as a triple $(X, D, \mathcal{C})$ where

- $X = \{x_0, \ldots, x_{n-1}\}$ is a set of variables
- D is a finite set, the *domain* for the variables
- $\mathcal{C} = \{(v_0, R_0), \ldots, (v_{n-1}, R_{n-1})\}$ is a set of *constraints*, i.e. $v_i \in X^{k_i}$ and $R_i \subseteq D^{k_i}$ for some $k_i > 0$

A *solution* of the CSP $(X, D, \mathcal{C})$ is a function $h : X \to D$ such that $\bar{h}(v_i) \in R_i$, where $\bar{h}(x_0, \ldots, x_{k-1}) = (h(x_0), \ldots, h(x_{k-1}))$

Instead use a Constraint Satisfaction solver, e.g. Minion (minion.sourceforge.net)

A *constraint satisfaction problem* (CSP) is traditionally given as a triple $(X, D, \mathcal{C})$ where

- $X = \{x_0, \ldots, x_{n-1}\}$ is a set of variables
- D is a finite set, the *domain* for the variables
- $\mathcal{C} = \{(v_0, R_0), \ldots, (v_{n-1}, R_{n-1})\}$ is a set of *constraints*, i.e. $v_i \in X^{k_i}$ and $R_i \subseteq D^{k_i}$ for some $k_i > 0$

A *solution* of the CSP $(X, D, \mathcal{C})$ is a function $h : X \to D$ such that $\bar{h}(v_i) \in R_i$, where $\bar{h}(x_0, \ldots, x_{k-1}) = (h(x_0), \ldots, h(x_{k-1}))$

Instead use a Constraint Satisfaction solver, e.g. Minion (minion.sourceforge.net)

A *constraint satisfaction problem* (CSP) is traditionally given as a triple $(X, D, \mathcal{C})$ where

- $X = \{x_0, \ldots, x_{n-1}\}$ is a set of variables
- D is a finite set, the *domain* for the variables
- $\mathcal{C} = \{(v_0, R_0), \ldots, (v_{n-1}, R_{n-1})\}$ is a set of *constraints*, i.e. $v_i \in X^{k_i}$ and $R_i \subseteq D^{k_i}$ for some $k_i > 0$

A *solution* of the CSP $(X, D, \mathcal{C})$ is a function $h : X \to D$ such that $\bar{h}(v_i) \in R_i$, where $\bar{h}(x_0, \ldots, x_{k-1}) = (h(x_0), \ldots, h(x_{k-1}))$

Instead use a Constraint Satisfaction solver, e.g. Minion (minion.sourceforge.net)

A *constraint satisfaction problem* (CSP) is traditionally given as a triple $(X, D, \mathcal{C})$ where

- $X = \{x_0, \ldots, x_{n-1}\}$ is a set of variables
- D is a finite set, the *domain* for the variables
- $\mathcal{C} = \{(v_0, R_0), \ldots, (v_{n-1}, R_{n-1})\}$ is a set of *constraints*, i.e. $v_i \in X^{k_i}$ and $R_i \subseteq D^{k_i}$ for some $k_i > 0$

A *solution* of the CSP $(X, D, \mathcal{C})$ is a function $h : X \to D$ such that $\bar{h}(v_i) \in R_i$, where $\bar{h}(x_0, \ldots, x_{k-1}) = (h(x_0), \ldots, h(x_{k-1}))$

Instead use a Constraint Satisfaction solver, e.g. Minion (minion.sourceforge.net)

A *constraint satisfaction problem* (CSP) is traditionally given as a triple $(X, D, \mathcal{C})$ where

- $X = \{x_0, \ldots, x_{n-1}\}$ is a set of variables
- D is a finite set, the *domain* for the variables
- $\mathcal{C} = \{(v_0, R_0), \ldots, (v_{n-1}, R_{n-1})\}$ is a set of *constraints*, i.e. $v_i \in X^{k_i}$ and $R_i \subseteq D^{k_i}$ for some $k_i > 0$

A *solution* of the CSP $(X, D, \mathcal{C})$ is a function $h : X \to D$ such that $\bar{h}(v_i) \in R_i$, where $\bar{h}(x_0, \ldots, x_{k-1}) = (h(x_0), \ldots, h(x_{k-1}))$

Instead use a Constraint Satisfaction solver, e.g. Minion (minion.sourceforge.net)

A *constraint satisfaction problem* (CSP) is traditionally given as a triple $(X, D, \mathcal{C})$ where

- $X = \{x_0, \ldots, x_{n-1}\}$ is a set of variables
- D is a finite set, the *domain* for the variables
- $\mathcal{C} = \{(v_0, R_0), \ldots, (v_{n-1}, R_{n-1})\}$ is a set of *constraints*, i.e. $v_i \in X^{k_i}$ and $R_i \subseteq D^{k_i}$ for some $k_i > 0$

A *solution* of the CSP $(X, D, \mathcal{C})$ is a function $h : X \to D$ such that $\bar{h}(v_i) \in R_i$, where $\bar{h}(x_0, \ldots, x_{k-1}) = (h(x_0), \ldots, h(x_{k-1}))$

The *generalized CSP* is a pair $(\mathbf{U}, \mathbf{V})$ of finite relational structures of the same type with finitely many relation symbols

A solution of the generalized CSP is a homomorphism $h : \mathbf{U} \to \mathbf{V}$

Lemma

The CSP and the generalized CSP are equivalent

Given $(X, D, \mathcal{C})$, let $\mathbf{U} = (X, \{v_0\}, \ldots, \{v_{m-1}\})$ and $\mathbf{V} = (D, R_0, \ldots, R_{m-1})$

Conversely, given a pair $(\mathbf{U}, \mathbf{V})$, both of type $R_0, \ldots, R_{t-1}$

let $X = U$, $D = V$ and $\mathcal{C} = \{(v, R_i^{\mathbf{V}}) | v \in R_i^{\mathbf{U}}, i = 0, \ldots, t-1\}$

The *generalized CSP* is a pair $(\mathbf{U}, \mathbf{V})$ of finite relational structures of the same type with finitely many relation symbols

A solution of the generalized CSP is a homomorphism $h : \mathbf{U} \to \mathbf{V}$

Lemma

The CSP and the generalized CSP are equivalent

Given $(X, D, \mathcal{C})$, let $\mathbf{U} = (X, \{v_0\}, \ldots, \{v_{m-1}\})$ and $\mathbf{V} = (D, R_0, \ldots, R_{m-1})$

Conversely, given a pair $(\mathbf{U}, \mathbf{V})$, both of type $R_0, \ldots, R_{t-1}$

let $X = U$, $D = V$ and $\mathcal{C} = \{(v, R_i^{\mathbf{V}}) | v \in R_i^{\mathbf{U}}, i = 0, \ldots, t-1\}$

The *generalized CSP* is a pair $(\mathbf{U}, \mathbf{V})$ of finite relational structures of the same type with finitely many relation symbols

A solution of the generalized CSP is a homomorphism $h : \mathbf{U} \to \mathbf{V}$

Lemma

The CSP and the generalized CSP are equivalent

Given $(X, D, \mathcal{C})$, let $\mathbf{U} = (X, \{v_0\}, \ldots, \{v_{m-1}\})$ and $\mathbf{V} = (D, R_0, \ldots, R_{m-1})$

Conversely, given a pair $(\mathbf{U}, \mathbf{V})$, both of type $R_0, \ldots, R_{t-1}$

let $X = U$, $D = V$ and $\mathcal{C} = \{(v, R_i^{\mathbf{V}}) | v \in R_i^{\mathbf{U}}, i = 0, \ldots, t-1\}$

The *generalized CSP* is a pair $(\mathbf{U}, \mathbf{V})$ of finite relational structures of the same type with finitely many relation symbols

A solution of the generalized CSP is a homomorphism $h : \mathbf{U} \to \mathbf{V}$

Lemma

The CSP and the generalized CSP are equivalent

Given $(X, D, \mathcal{C})$, let $\mathbf{U} = (X, \{v_0\}, \ldots, \{v_{m-1}\})$ and $\mathbf{V} = (D, R_0, \ldots, R_{m-1})$

Conversely, given a pair $(\mathbf{U}, \mathbf{V})$, both of type $R_0, \ldots, R_{t-1}$

let $X = U$, $D = V$ and $\mathcal{C} = \{(v, R_i^{\mathbf{V}}) | v \in R_i^{\mathbf{U}}, i = 0, \ldots, t-1\}$

The *generalized CSP* is a pair $(\mathbf{U}, \mathbf{V})$ of finite relational structures of the same type with finitely many relation symbols

A solution of the generalized CSP is a homomorphism $h : \mathbf{U} \to \mathbf{V}$

Lemma

The CSP and the generalized CSP are equivalent

Given $(X, D, \mathcal{C})$, let $\mathbf{U} = (X, \{v_0\}, \ldots, \{v_{m-1}\})$ and $\mathbf{V} = (D, R_0, \ldots, R_{m-1})$

Conversely, given a pair $(\mathbf{U}, \mathbf{V})$, both of type $R_0, \ldots, R_{t-1}$

let $X = U$, $D = V$ and $\mathcal{C} = \{(v, R_i^{\mathbf{V}}) | v \in R_i^{\mathbf{U}}, i = 0, \ldots, t-1\}$

The *generalized CSP* is a pair $(\mathbf{U}, \mathbf{V})$ of finite relational structures of the same type with finitely many relation symbols

A solution of the generalized CSP is a homomorphism $h : \mathbf{U} \to \mathbf{V}$

Lemma

The CSP and the generalized CSP are equivalent

Given $(X, D, \mathcal{C})$, let $\mathbf{U} = (X, \{v_0\}, \ldots, \{v_{m-1}\})$ and $\mathbf{V} = (D, R_0, \ldots, R_{m-1})$

Conversely, given a pair $(\mathbf{U}, \mathbf{V})$, both of type $R_0, \ldots, R_{t-1}$

let $X = U$, $D = V$ and $\mathcal{C} = \{(v, R_i^{\mathbf{V}}) | v \in R_i^{\mathbf{U}}, i = 0, \ldots, t-1\}$

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: Hom($\mathbf{A}, \mathbf{B}$), End($\mathbf{A}$),

Emb($\mathbf{A}, \mathbf{B}$), Aut($\mathbf{A}$), $\mathrm{Pol}_1(\mathbf{A})$

is_hom_imgage($\mathbf{A}, \mathbf{B}$), is_subalgebra($\mathbf{A}, \mathbf{B}$), is_isomorphic($\mathbf{A}, \mathbf{B}$)

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: Hom($\mathbf{A}, \mathbf{B}$), End($\mathbf{A}$),

Emb($\mathbf{A}, \mathbf{B}$), Aut($\mathbf{A}$), $\mathrm{Pol}_1(\mathbf{A})$

is_hom_imgage($\mathbf{A}, \mathbf{B}$), is_subalgebra($\mathbf{A}, \mathbf{B}$), is_isomorphic($\mathbf{A}, \mathbf{B}$)

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: Hom($\mathbf{A}, \mathbf{B}$), End($\mathbf{A}$),

Emb($\mathbf{A}, \mathbf{B}$), Aut($\mathbf{A}$), $\mathrm{Pol}_1(\mathbf{A})$

is_hom_imgage($\mathbf{A}, \mathbf{B}$), is_subalgebra($\mathbf{A}, \mathbf{B}$), is_isomorphic($\mathbf{A}, \mathbf{B}$)

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: $\mathrm{Hom}(\mathbf{A}, \mathbf{B})$, $\mathrm{End}(\mathbf{A})$,

$\mathrm{Emb}(\mathbf{A}, \mathbf{B})$, $\mathrm{Aut}(\mathbf{A})$, $\mathrm{Pol}_1(\mathbf{A})$

is_hom_imgage($\mathbf{A}, \mathbf{B}$), is_subalgebra($\mathbf{A}, \mathbf{B}$), is_isomorphic($\mathbf{A}, \mathbf{B}$)

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: Hom($\mathbf{A}, \mathbf{B}$), End($\mathbf{A}$),

Emb($\mathbf{A}, \mathbf{B}$), Aut($\mathbf{A}$), $\mathrm{Pol}_1(\mathbf{A})$

is_hom_imgage($\mathbf{A}, \mathbf{B}$), is_subalgebra($\mathbf{A}, \mathbf{B}$), is_isomorphic($\mathbf{A}, \mathbf{B}$)

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: $\mathrm{Hom}(\mathbf{A}, \mathbf{B})$, $\mathrm{End}(\mathbf{A})$,

$\mathrm{Emb}(\mathbf{A}, \mathbf{B})$, $\mathrm{Aut}(\mathbf{A})$, $\mathrm{Pol}_1(\mathbf{A})$

is_hom_imgage$(\mathbf{A}, \mathbf{B})$, is_subalgebra$(\mathbf{A}, \mathbf{B})$, is_isomorphic$(\mathbf{A}, \mathbf{B})$

CSP solvers have been developed for the past 30 years

Minion is a recent open-source CSP solver

Input syntax is similar to the traditional $(X, D, \mathcal{C})$ version

Wrote short Sage/Python routines that convert a pair of finite algebras into a Minion file, call Minion and read the output file back into Sage

Implemented commands: Hom$(\mathbf{A}, \mathbf{B})$, End$(\mathbf{A})$,

Emb$(\mathbf{A}, \mathbf{B})$, Aut$(\mathbf{A})$, Pol$_1(\mathbf{A})$

is_hom_imgage$(\mathbf{A}, \mathbf{B})$, is_subalgebra$(\mathbf{A}, \mathbf{B})$, is_isomorphic$(\mathbf{A}, \mathbf{B})$

The number of subdirectly irreducible lattices of size n:

n	All	SI
1	1	0
2	1	1
3	1	0
4	2	0
5	5	2
6	15	4
7	53	16
8	222	69
9	1078	360
10	5994	2103
11	37622	13867
12	262776	100853

Heitzig and Reinhold [2002] enumerated all lattices up to $n = 18$

The number of subdirectly irreducible lattices of size n:

n	All	SI
1	1	0
2	1	1
3	1	0
4	2	0
5	5	2
6	15	4
7	53	16
8	222	69
9	1078	360
10	5994	2103
11	37622	13867
12	262776	100853

Heitzig and Reinhold [2002] enumerated all lattices up to $n = 18$

The number of subdirectly irreducible lattices of size n:

n	All	SI
1	1	0
2	1	1
3	1	0
4	2	0
5	5	2
6	15	4
7	53	16
8	222	69
9	1078	360
10	5994	2103
11	37622	13867
12	262776	100853

Heitzig and Reinhold [2002] enumerated all lattices up to $n = 18$

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

There are 2555 subdirectly irreducible lattices of size ≤ 10

The poset of subvarieties that they generate has height 7

The bottom (level 0) is the variety of distributive lattices

Level 1 has two varieties generated by M_3 and N_5 found by Dedekind, 1900

Level 2 has 25 varieties:

2 modular M_4 and $M_{3,3}$, found by Grätzer 1966, Jónsson 1968

15 nonmodular $L_1, \ldots, L_{15}$ covering N_5, found by McKenzie 1972

8 covering both M_3 and N_5, found by Ruckelshausen 1978, 1983

Level 3 has not yet been fully investigated

H Rose [84], JG Lee [85], JB Nation [85,86,90], CY Wong [89] found all join-irreducible covers of $L_1, \ldots, L_{15}$

But there are many join-reducible covers with join-irreducibles above them

E.g. there is a join-irreducible variety above L_1, L_2, L_4, L_5 that is generated by the 6-element fence plus top and bottom

Level 3 has not yet been fully investigated

H Rose [84], JG Lee [85], JB Nation [85,86,90], CY Wong [89] found all join-irreducible covers of $L_1, \ldots, L_{15}$

But there are many join-reducible covers with join-irreducibles above them

E.g. there is a join-irreducible variety above L_1, L_2, L_4, L_5 that is generated by the 6-element fence plus top and bottom

Level 3 has not yet been fully investigated

H Rose [84], JG Lee [85], JB Nation [85,86,90], CY Wong [89] found all join-irreducible covers of $L_1, \ldots, L_{15}$

But there are many join-reducible covers with join-irreducibles above them

E.g. there is a join-irreducible variety above L_1, L_2, L_4, L_5 that is generated by the 6-element fence plus top and bottom

Level 3 has not yet been fully investigated

H Rose [84], JG Lee [85], JB Nation [85,86,90], CY Wong [89] found all join-irreducible covers of $L_1, \ldots, L_{15}$

But there are many join-reducible covers with join-irreducibles above them

E.g. there is a join-irreducible variety above L_1, L_2, L_4, L_5 that is generated by the 6-element fence plus top and bottom

Other computational results

Using Birkhoff's duality between complete perfect complemented lattices and separated graphs

enumerated finite complemented lattices with up to 8 join-irreducibles (includes the BA with 256 elements)

Computed levels of subvarieties of bi-semilattices (non-congruence distributive) using the $A \in \mathrm{Var}(B)$ implementation of the UA Calculator

Other computational results

Using Birkhoff's duality between complete perfect complemented lattices and separated graphs

enumerated finite complemented lattices with up to 8 join-irreducibles (includes the BA with 256 elements)

Computed levels of subvarieties of bi-semilattices (non-congruence distributive) using the $A \in \mathrm{Var}(B)$ implementation of the UA Calculator

Other computational results

Using Birkhoff's duality between complete perfect complemented lattices and separated graphs

enumerated finite complemented lattices with up to 8 join-irreducibles (includes the BA with 256 elements)

Computed levels of subvarieties of bi-semilattices (non-congruence distributive) using the $A \in \mathrm{Var}(B)$ implementation of the UA Calculator

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \leq 1 \leq xx^l$ and $xx^r \leq 1 \leq x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \leq 1 \leq xx^l$ and $xx^r \leq 1 \leq x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \leq 1 \leq xx^l$ and $xx^r \leq 1 \leq x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \leq 1 \leq xx^l$ and $xx^r \leq 1 \leq x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \le 1 \le x x^l$ and $x x^r \le 1 \le x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \le 1 \le xx^l$ and $xx^r \le 1 \le x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

Prover9/Mace4 (prover9.org) is a first-order theorem prover and countermodel finder

Waldmeister is an equational theorem prover

Both were used to investigate ℓ-pregroups, i.e. lattice-ordered monoids with two unary operations x^l, x^r that satisfy $x^l x \leq 1 \leq x x^l$ and $x x^r \leq 1 \leq x^r x$

An ℓ-pregroup is *periodic* if it satisfies the identity $x^{l^n} = x^{r^n}$ for some positive integer n

Theorem (N. Galatos, P. J. 2010)

Periodic ℓ-pregroups have distributive lattice reducts

Parts of the proof were found with the equational theorem prover Waldmeister (273 lemmas, $>$ 140 pages pdf)

Fortunately it was possible to condense the proof down to 3 pages

sagemath.org

uacalc.org

minion.sourceforge.net

prover9.org

Thank You