



HAL
open science

Counting and Computing Join-Endomorphisms in Lattices

Santiago Quintero, Sergio Ramírez, Camilo Rueda, Frank D. Valencia

► **To cite this version:**

Santiago Quintero, Sergio Ramírez, Camilo Rueda, Frank D. Valencia. Counting and Computing Join-Endomorphisms in Lattices. 2019. hal-02422624v1

HAL Id: hal-02422624

<https://hal.science/hal-02422624v1>

Preprint submitted on 22 Dec 2019 (v1), last revised 19 Dec 2023 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Counting and Computing Join-Endomorphisms in Lattices

Santiago Quintero², Sergio Ramirez¹, Camilo Rueda¹, Frank Valencia^{1,2}

¹ Pontificia Universidad Javeriana Cali

² CNRS & LIX, École Polytechnique de Paris

Abstract. Structures involving a lattice and join-endomorphisms on it are ubiquitous in computer science. We study the cardinality of the set $\mathfrak{J}(L)$ of all join-endomorphisms of a given finite lattice L . We show that the cardinality of $\mathfrak{J}(L)$ is sub-exponential, exponential and super-exponential in the size of the lattice for boolean algebras, linear-orders, and arbitrary lattices, respectively. We also study the following problem: Given a lattice L of size n and a set $S \subseteq \mathfrak{J}(L)$ of size m , find the greatest lower bound $\prod_{\mathfrak{J}(L)} S$. The join-endomorphism $\prod_{\mathfrak{J}(L)} S$ has meaningful interpretations in epistemic logic, distributed systems, and Aumann structures. We show that this problem can be solved with worst-case time complexity in $O(mn^{\log_2 3})$ for powerset lattices, $O(mn^2)$ for lattices of sets, and $O(mn + n^3)$ for arbitrary lattices. The complexity is expressed in terms of the basic binary lattice operations performed by the algorithm.

1 Introduction

There is a long established tradition of using lattices to model structural entities in many fields of mathematics and computer science. For example, lattices are used in concurrency theory to represent the hierarchical organization of the information resulting from agent's interactions [10]. *Mathematical morphology* (MM), a well-established theory for the analysis and processing of geometrical structures, is founded upon lattice theory [1,11]. Lattices are also used as algebraic structures for modal and epistemic logics as well as Aumann structures (e.g., modal algebras and constraint systems [7]).

In all these and many other applications, lattice *join-endomorphisms* appear as fundamental. In MM, join-endomorphisms correspond to one of its fundamental operations; *dilations*. In modal algebra, they correspond via duality to the box modal operator. In epistemic settings, they represent belief or knowledge of agents. In fact, our own interest in lattice theory derives from using join-endomorphisms to model the perception that agents may have of a statement in a lattice of partial information [7].

For finite lattices, devising suitable algorithms to compute lattice maps with some given properties would thus be of great utility. We are interested in constructing algorithms for computing lattice morphisms. This requires, first, a careful study of the space of such maps to have a clear idea of how particular lattice structures impact on the size of the space. We are, moreover, particularly interested in computing the *maximum* join-endomorphism below a given collection of join-morphisms. This turns out to be important, among others, in spatial computation (and in epistemic logic) to model the distributed information (resp. distributed knowledge) available to a set of agents as

conforming a group [8]. It could also be regarded as the maximum perception consistent with (or derivable from) a collection of perceptions of a group of agents.

Problem. Consider the set $\mathfrak{J}(L)$ of all join-endomorphisms of a finite lattice L . The set $\mathfrak{J}(L)$ can be made into a lattice by ordering join-endomorphisms point-wise wrt the order of L . We investigate the following maximization problem: *Given a lattice L of size n and a set $S \subseteq \mathfrak{J}(L)$ of size m , find in $\mathfrak{J}(L)$ the greatest lower bound of S , i.e., $\prod_{\mathfrak{J}(L)} S$. Simply taking $\sigma : L \rightarrow L$ with $\sigma(e) \stackrel{\text{def}}{=} \prod_L \{f(e) \mid f \in S\}$ does not solve the problem as σ may not be a join-endomorphism. Furthermore, since $\mathfrak{J}(L)$ can be seen as the search space, we also consider the problem of determining its cardinality. Our main results are the following.*

Contributions. We give pleasant characterizations of the exact cardinality of $\mathfrak{J}(L)$ for some fundamental lattices in terms of sub-exponentials, the central binomial coefficient, and Laguerre polynomials. We show that the number of join-endomorphisms is **(1)** *sub-exponential* $n^{\log_2 n}$ for powerset lattices, **(2)** *exponential* $\binom{2n}{n}$ for linear lattices. We also consider the stereotypical non-distributive lattice \mathbf{M}_n . We show that **(3)** $|\mathfrak{J}(\mathbf{M}_n)|$ equals the *super-exponential* $r_0^n + \dots + r_n^n + r_1^{n+1} = n! \mathcal{L}_n(-1) + (n+1)^2$. (r_k^n is number of ways to place k non-attacking rooks on an $m \times m$ board and $\mathcal{L}_n(x)$ is the Laguerre polynomial of degree n .) Furthermore, **(4)** we provide an algorithm that computes $\prod_{\mathfrak{J}(L)}(S)$ with worst-case time complexity in $O(mn^{\log_2 3})$ for powerset lattices and $O(mn^2)$ for lattices of sets. For general lattices, **(5)** we provide an algorithm that computes $\prod_{\mathfrak{J}(L)}(S)$ with worst-case time complexity in $O(nm + n^3)$.

Organization. Section 2 introduces the background. In Section 3 we present the contributions **(1-3)** regarding the cardinality of $\mathfrak{J}(L)$. In Section 4 we provide the algorithms for computing $\prod_{\mathfrak{J}(L)}(S)$ from contributions **(4-5)**. We present experiments and an example in Section 4.5. Due to space restrictions, some proofs are given in the Appendix.

2 Background: Join-Endomorphisms and Their Space

We presuppose basic knowledge of order theory [2] and use the following notions. Let (L, \sqsubseteq) be a partially orderer set (poset), and let $S \subseteq L$. We use $\bigsqcup_L S$ to denote the least upper bound (or *supremum* or *join*) of S in L , if it exists. Dually, $\prod_L S$ is the greatest lower bound (glb) (*infimum* or *meet*) of S in L , if it exists. We shall often omit the index L from \bigsqcup_L and \prod_L when no confusion arises. As usual, if $S = \{c, d\}$, $c \sqcup d$ and $c \sqcap d$ represent $\bigsqcup S$ and $\prod S$, respectively. If L has a greatest element (top) \top , and a least element (bottom) \perp , we have $\bigsqcup \emptyset = \perp$ and $\prod \emptyset = \top$. The poset L is *distributive* iff for every $a, b, c \in L$, $a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$.

The poset L is a *lattice* iff each finite subset of L has a supremum and infimum in L , and it is a *complete lattice* iff each subset of L has a supremum and infimum in L .

A *self-map* on L is a function $f : L \rightarrow L$. A self-map f is *monotonic* if $a \sqsubseteq b$ implies $f(a) \sqsubseteq f(b)$, and f *preserves* the join of $S \subseteq L$ iff $f(\bigsqcup S) = \bigsqcup \{f(c) \mid c \in S\}$.

We shall use the following posets and notation. Given n , we use \mathbf{n} to denote the poset $\{1, \dots, n\}$ with the linear order $x \sqsubseteq y$ iff $x \leq y$. The poset $\bar{\mathbf{n}}$ is the set $\{1, \dots, n\}$ with the discrete order $x \sqsubseteq y$ iff $x = y$. Given a poset L , we use L_\perp for the poset that results from adding a bottom element to L . The poset L^\top is similarly defined. The

lattice $\mathbf{2}^n$ is the n -fold Cartesian product of $\mathbf{2}$ ordered coordinate-wise. We define \mathbf{M}_n as the lattice $(\mathbf{n}_\perp)^\top$. A *lattice of sets* is a set of sets ordered by inclusion and closed under finite unions and intersections. A *powerset lattice* is a lattice of sets that includes all the subsets of its top element.

We shall investigate the set of all join-endomorphisms of a given lattice ordered point-wise. Notice that every finite lattice is complete lattice.

Definition 1 (Join-endomorphisms and their space). *Let L be a complete lattice. We say that a self-map is a (lattice) join-endomorphism iff it preserves the join of every finite subset of L . Define $\mathfrak{J}(L)$ as the set of all join-endomorphisms of L . Furthermore, given $f, g \in \mathfrak{J}(L)$, define $f \sqsubseteq_{\mathfrak{J}} g$ iff $f(a) \sqsubseteq g(a)$ for every $a \in L$.*

The following are immediate consequences of the above definition.

Proposition 1. *Let L be a complete lattice. $f \in \mathfrak{J}(L)$ iff $f(\perp) = \perp$ and $f(a \sqcup b) = f(a) \sqcup f(b)$ for all $a, b \in L$. If f is a join-endomorphism of L then f is monotonic.*

Given a set $S \subseteq \mathfrak{J}(L)$, where L is a finite lattice, we are interested in finding the greatest join-endomorphism in $\mathfrak{J}(L)$ below the elements of S , i.e., $\prod_{\mathfrak{J}(L)} S$. Since every finite lattice is also a complete lattice, the existence of $\prod_{\mathfrak{J}(L)} S$ is guaranteed by the following proposition from [6].

Proposition 2. *Let (L, \sqsubseteq) be a complete lattice. Then $(\mathfrak{J}(L), \sqsubseteq_{\mathfrak{J}})$ is a complete lattice.*

In the following sections we study join-endomorphisms of some fundamental families of finite lattices. In particular, powerset lattices and lattices of sets. Birkhoff's representation theorem bear witness to the importance of these lattices: *Every finite distributive lattice is isomorphic to a lattice of sets and any finite boolean algebra is isomorphic to a powerset lattice* [2]. We shall investigate the cardinality of $\mathfrak{J}(L)$ and also provide efficient algorithms to compute $\prod_{\mathfrak{J}(L)} S$.

3 The Size of the Function Space

In this section we determine the size of $\mathfrak{J}(L)$ for powerset lattices, and for two extreme opposites; the total order and the discrete order extended with a top and bottom.

3.1 Distributed Lattices

We begin with lattices isomorphic to $\mathbf{2}^n$. They include *finite boolean algebras* and *powerset lattices* [2].

Theorem 1. *Suppose that $m \geq 0$. Let L be any lattice isomorphic to the product lattice $\mathbf{2}^m$. Then $|\mathfrak{J}(L)| = n^{\log_2 n}$ where $n = 2^m$ is the size of L .*

Thus powerset lattices and boolean algebras have a super-polynomial, sub-exponential number of join-endomorphisms. Nevertheless, linear order lattices allows for an exponential number of join-endomorphisms given by the *central binomial coefficient*.

Theorem 2. *Suppose that $n \geq 0$. Let L be any lattice isomorphic to the linear order lattice \mathbf{n}_\perp . Then $|\mathfrak{J}(L)| = \binom{2n}{n}$.*

It is easy to prove that $\frac{1}{\sqrt{\pi(n+\frac{1}{2})}} 4^n \leq \binom{2n}{n} \leq 4^n$ for $n \geq 1$. Together with Thm.2, this gives us explicit exponential lower and upper bounds for $|\mathfrak{J}(L)|$ for linear lattices.

3.2 Non-distributive Case

For general lattices, the number of join-endomorphisms can be super-exponential and be characterized in terms of Laguerre (and rook) polynomials.

Laguerre polynomials are solutions to Laguerre's second-order linear differential equation $xy'' + (1-x)y' + ny = 0$. The Laguerre polynomial of degree n in x , $\mathcal{L}_n(x)$ is given by the summation $\sum_{k=0}^n \binom{n}{k} \frac{(-1)^k}{k!} x^k$.

The lattice \mathbf{M}_n is non-distributive for any $n \geq 3$. The size of $\mathfrak{J}(\mathbf{M}_n)$ can be succinctly expressed as follows.

Theorem 3. $|\mathfrak{J}(\mathbf{M}_n)| = (n+1)^2 + n!\mathcal{L}_n(-1)$.

In combinatorics rook polynomials are generating functions of the number of ways to place non-attacking rooks on a board. A *rook polynomial* (for square boards) $\mathcal{R}_n(x)$ has the form $\sum_{k=0}^n x^k r(k, n)$ where the (rook) coefficient $r(k, n)$ represents the number of ways to place k non-attacking rooks on an $n \times n$ chessboard. For example, $r(0, n) = 1$, $r(1, n) = n^2$, $r(n, n) = n!$. In general $r(k, n) = \binom{n}{k}^2 k!$.

Rook polynomials are related to Laguerre polynomials by $\mathcal{R}_n(x) = n!x^n \mathcal{L}_n(-x^{-1})$. Therefore, as a direct consequence of the above theorem, we can also characterize $|\mathfrak{J}(\mathbf{M}_n)|$ in combinatorial terms as the following sum of rook coefficients.

Corollary 1. *Let $r'(n+1, n) = r(1, n+1)$ and $r'(k, n) = r(k, n)$ if $k \leq n$. Then $|\mathfrak{J}(\mathbf{M}_n)| = \sum_{k=0}^{n+1} r'(k, n)$.*

We conclude this section with another pleasant correspondence between the endomorphisms in $\mathfrak{J}(\mathbf{M}_n)$ and $\mathcal{R}_n(x)$. Let $f : L \rightarrow L$ be a function over a lattice (L, \sqsubseteq) . We say that f is *non-reducing* in L iff it does not map any value to a smaller one; i.e., there is no $e \in L$ such that $f(e) \sqsubset e$. The number of join-endomorphisms that are non-reducing in \mathbf{M}_n is exactly the value of the rook polynomial $\mathcal{R}_n(x)$ for $x = 1$.

Theorem 4. $\mathcal{R}_n(1) = |\{ f \in \mathfrak{J}(\mathbf{M}_n) \mid f \text{ is non-reducing in } \mathbf{M}_n \}|$.

We now present the proofs of Thm.1-3, the proof of Thm.4 is given in the Appendix A.

Proof of Theorem 1. We wish to prove the following: *Let L be any lattice isomorphic to the product lattice $\mathbf{2}^m$. Then $|\mathfrak{J}(L)| = n^{\log_2 n}$ where $n = 2^m$ is the size of L .*

Take any set S of size m . Consider the powerset lattice $L = \mathcal{P}(S)$ ordered by inclusion. We have $n = |\mathcal{P}(S)| = 2^m$. We shall show that $|\mathfrak{J}(L)| = n^{\log_2 n}$. Since $\mathcal{P}(S)$ is isomorphic to $\mathbf{2}^m$, Theorem 1 follows from the fact that isomorphic lattices have the same number of join-endomorphisms.

Let \mathcal{F} be the family of functions $f : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$ that satisfy (a) $f(T) = \bigcup_{t \in T} f(\{t\})$ if $|T| > 1$ and (b) $f(\emptyset) = \emptyset$. The equality $|\mathfrak{J}(L)| = n^{\log_2 n}$ follows from the following claim: (1) $\mathcal{F} = \mathfrak{J}(L)$ and (2) $|\mathcal{F}| = n^{\log_2 n}$.

To prove (1) one can verify that if $f \in \mathcal{F}$ then f is a join-endomorphism where \sqcup is \cup and \perp is the \emptyset . Hence $f \in \mathfrak{J}(L)$. On the other hand, if $f \notin \mathcal{F}$ then either $f(T) \neq \bigcup_{t \in T} f(\{t\})$ for some $T \subseteq S$ or $f(\emptyset) \neq \emptyset$. But since $\sqcup = \cup$ and $\perp = \emptyset$, we have $T = \bigsqcup_{t \in T} \{t\}$ but $f(T) \neq \bigsqcup_{t \in T} f(\{t\})$ or $f(\perp) \neq \perp$. Hence $f \notin \mathfrak{J}(L)$.

To prove (2) notice that given $f \in \mathcal{F}$, for each $T \subseteq S$ if $|T| > 1$ then the value $f(T)$ is determined by the values of f applied to each singleton $\{t\} \subseteq S$, and if $|T| = 0$ the value $f(T)$ is fixed to \emptyset . The set $\mathcal{P}(S)$ has $\log_2 n = m$ singletons. Since there is no restriction on how each $f \in \mathcal{F}$ should map singletons, $|\mathcal{F}| = n^{\log_2 n}$ as wanted. \square

Proof of Theorem 2. We now show that the size of $\mathfrak{J}(L)$ for linear orders is determined by the central binomial coefficient. Let L be any lattice isomorphic to the linear order lattice \mathbf{n}_\perp . We want to show that $|\mathfrak{J}(L)| = \binom{2n}{n}$.

Let $\mathcal{M}_\perp(L)$ be the set of monotonic functions from L to L that preserve \perp . We claim that $\mathcal{M}_\perp(L) = \mathfrak{J}(L)$. The inclusion $\mathfrak{J}(L) \subseteq \mathcal{M}_\perp(L)$ follows from Proposition 1 and the fact that join-endomorphisms preserve bottoms. For $\mathcal{M}_\perp(L) \subseteq \mathfrak{J}(L)$, take $f \in \mathcal{M}_\perp(L)$. By definition $f(\perp) = \perp$. Take any $a, b \in L$. So either $a \sqsubseteq b$ or $b \sqsubseteq a$. If $a \sqsubseteq b$ then $f(a \sqcup b) = f(b)$ and by monotonicity of f , $f(b) = f(a) \sqcup f(b)$. Similarly if $b \sqsubseteq a$ then $f(a \sqcup b) = f(a) = f(a) \sqcup f(b)$. We conclude that $f \in \mathfrak{J}(L)$.

Now, for every $f \in \mathcal{M}_\perp(L)$ we have $f(\perp) = \perp$, then $|\mathfrak{J}(L)| = |\mathcal{M}_\perp(L)| = |\mathcal{M}(L \setminus \{\perp\} \rightarrow L)|$ where $\mathcal{M}(L \setminus \{\perp\} \rightarrow L)$ is the set of monotonic functions from $L \setminus \{\perp\}$ to L . Thus to prove Theorem 2 it suffices to show $|\mathcal{M}(L \setminus \{\perp\} \rightarrow L)| = \binom{2n}{n}$.

Notice that $|L| = n + 1$. Consider the equation $\sum_{i=1}^{n+1} X_i = n$ where the variable X_i takes a value between 0 and n . Let $Sol(n)$ be the set of all solutions to this equation. We can show that $|Sol(n)| = |\mathcal{M}(L \setminus \{\perp\} \rightarrow L)|$ by providing the following bijection $\sigma : \mathcal{M}(L \setminus \{\perp\} \rightarrow L) \rightarrow Sol(n)$. The function σ associates each $f \in \mathcal{M}(L \setminus \{\perp\} \rightarrow L)$ with a solution $\sigma(f)$ assigning to every X_i the number of consecutive values from $L \setminus \{\perp\}$ mapped by f to the i -th value of L .

From combinatorics we know that for any pair of positive integers n and k , the number of k -tuples of non-negative integers whose sum is n equals $\binom{n+k-1}{n}$ [4]. For $k = n + 1$ these tuples correspond exactly to the solutions in $Sol(n)$. Therefore we have shown $|\mathfrak{J}(L)| = |\mathcal{M}_\perp(L)| = |\mathcal{M}(L \setminus \{\perp\} \rightarrow L)| = |Sol(n)| = \binom{2n}{n}$ as wanted. \square

Proof of Theorem 3. We show that $|\mathfrak{J}(\mathbf{M}_n)|$ is super-exponential and can be expressed in terms of Laguerre polynomials: $|\mathfrak{J}(\mathbf{M}_n)| = (n + 1)^2 + n! \mathcal{L}_n(-1)$.

Let $\mathcal{F} = \bigcup_{i=1}^4 \mathcal{F}_i$ where the mutually exclusive \mathcal{F}_i 's are defined in Table 1, and $I = \{1, \dots, n\}$. The proof is divided in two parts: (I) $\mathcal{F} = \mathfrak{J}(\mathbf{M}_n)$ and (II) $|\mathcal{F}| = (n + 1)^2 + n! \mathcal{L}_n(-1)$.

Part (I) For $\mathcal{F} \subseteq \mathfrak{J}(\mathbf{M}_n)$, it is easy to verify that each $f \in \mathcal{F}$ is a join-endomorphism.

For $\mathfrak{J}(\mathbf{M}_n) \subseteq \mathcal{F}$ we show that for any function f from \mathbf{M}_n to \mathbf{M}_n if $f \notin \mathcal{F}$, then $f \notin \mathfrak{J}(\mathbf{M}_n)$. Immediately, if $f(\perp) \neq \perp$ then $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Suppose $f(\perp) = \perp$. Let J, K, H be disjoint possibly empty sets s.t. $I = J \cup K \cup H$ and let $j = |J|$, $k = |K|$ and $h = |H|$. The sets J, K, H represent the elements of I mapped by f to \top , to elements of I , and to \perp , respectively. More precisely, $\text{Img}(f|_J) = \{\top\}$, $\text{Img}(f|_K) \subseteq I$ and $\text{Img}(f|_H) = \{\perp\}$. Furthermore, for every f either (1) $f(\top) = \perp$, (2) $f(\top) \in I$ or (3) $f(\top) = \top$. We show that $f \notin \mathfrak{J}(\mathbf{M}_n)$ for case (3), proofs of cases (1) and (2) are included in the Appendix A.

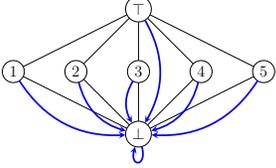
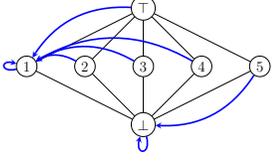
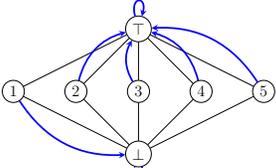
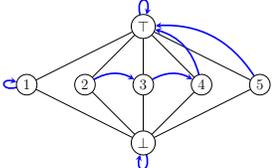
 <p>Let \mathcal{F}_1 be the family of functions f that for all $e \in \mathbf{M}_n$, $f(e) = \perp$.</p>	 <p>Let \mathcal{F}_2 be the family of bottom preserving functions f such that for some $e, e' \in I$: (a) $f(\top) = e$, (b) $f(e') = \perp$ or $f(e') = e$, and (c) $f(e'') = e$ for all $e'' \in I \setminus \{e'\}$.</p>
 <p>Let \mathcal{F}_3 be the family of top and bottom preserving functions f such that for some $e \in I$: (a) $f(e) = \perp$, and (b) $f(e') = \top$ for all $e' \in I \setminus \{e\}$.</p>	 <p>Let \mathcal{F}_4 be the family of top and bottom preserving functions where for each $f \in \mathcal{F}_4$ there are disjoint sets J, K_1, K_2 with $I = J \cup K_1 \cup K_2$ s.t.: (a) for all $e \in J$, $f(e) = \top$, (b) $f _{K_1}$ and $f _{K_2}$ are injective, and (c) $\text{Img}(f _{K_1}) \subseteq J$ and $\text{Img}(f _{K_2}) \subseteq I \setminus J$.</p>

Table 1: Families $\mathcal{F}_1, \dots, \mathcal{F}_4$ of join-endomorphisms of \mathbf{M}_n . $I = \{1, \dots, n\}$. $f|_A$ is the restriction of f to a subset A of its domain. $\text{Img}(f)$ is the image of f . A function from each \mathcal{F}_i for \mathbf{M}_5 is depicted with blue arrows.

Suppose $k = 0$. Notice that $f \notin \mathcal{F}_3$ and $f \notin \mathcal{F}_4$ hence $h \neq 1$ and $h \neq 0$. Thus $h > 1$ implies that there are at least two $e_1, e_2 \in H$ s.t. $f(e_1) = f(e_2) = \perp$. But then $f(e_1 \sqcup e_2) = f(\top) = \top \neq \perp = f(e_1) \sqcup f(e_2)$, hence $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Suppose $k > 0$. Assume $h = 0$. Let K_1, K_2 be disjoint possibly empty sets such that $K_1 \cup K_2 = K$, $\text{Img}(f|_{K_1}) \subseteq J$ and $\text{Img}(f|_{K_2}) \subseteq I \setminus J$. Notice that f is a \perp and \top preserving function and satisfies conditions (a) and (c) of \mathcal{F}_4 but $f \notin \mathcal{F}_4$, then f must violate condition (b). Thus $f|_{K_1}$ or $f|_{K_2}$ is not injective. Let us assume that $f|_{K_1}$ is not injective (the case when $f|_{K_2}$ is not injective is analogous). Then there are $a, b \in K_1$ and $c \in J$, such that $a \neq b$ but $f(a) = f(b) = c$. By definition $\top \notin J$, then $f(a) \sqcup f(b) = c \neq \top = f(a \sqcup b)$. Consequently, $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Assume $h > 0$. There must be $e_1, e_2, e_3 \in I$ such that $f(e_1) = \perp$ and $f(e_2) = e_3$. Notice that $f(e_1) \sqcup f(e_2) = e_3 \neq \top = f(\top) = f(e_1 \sqcup e_2)$. Therefore, $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Part (II) We prove that $|\mathcal{F}| = \sum_{i=1}^4 |\mathcal{F}_i| = (n+1)^2 + n!\mathcal{L}_n(-1)$. Recall that $n = |I|$. It is easy to prove that $|\mathcal{F}_1| = 1$, $|\mathcal{F}_2| = n^2 + n$ and $|\mathcal{F}_3| = n$. The reader is referred to Appendix A for details. Here we prove that $|\mathcal{F}_4| = n!\mathcal{L}_n(-1)$.

Let $f \in \mathcal{F}_4$ and let J, K_1, K_2 be disjoint possibly empty sets such that $I = J \cup K_1 \cup K_2$, $\text{Img}(f|_J) = \{\top\}$, $\text{Img}(f|_{K_1}) \subseteq J$ and $\text{Img}(f|_{K_2}) \subseteq I \setminus J$, where $f|_{K_1}$ and $f|_{K_2}$ are injective functions. We shall call $j = |J|$ and $k_1 = |K_1|$.

For each of the $\binom{n}{j}$ possibilities for J , the elements of K_1 are to be mapped to them by the injective function $f|_{K_1}$. This is possible only if $k_1 \leq j$, so we take $k_1 \leq \min(j, n-j)$. The number of choices of K_1 is $\binom{n-j}{k_1}$ and the number of choices among J that can be targets of those is $\binom{j}{k_1}$. Each of these can be mapped from any permutation of K_1 , so we have $\binom{j}{k_1} k_1! = \frac{j!}{(j-k_1)!}$ possibilities to choose K_1 and map its elements by $f|_{K_1}$ to elements of J . Similarly, the number of possibilities to choose K_2 and map its elements by $f|_{K_2}$ to elements of $I \setminus J$ is $\binom{n-j}{n-j-k_1} (n-j-k_1)! = \frac{(n-j)!}{k_1!}$. Therefore, $|\mathcal{F}_4| = \sum_{j=0}^n \binom{n}{j} \sum_{k_1=0}^{\min(j, n-j)} \binom{n-j}{k_1} \frac{j!}{(j-k_1)!} \frac{(n-j)!}{k_1!}$. We can simplify the inner sum: $\sum_{k_1=0}^{\min(j, n-j)} \binom{n-j}{k_1} \frac{j!}{(j-k_1)!} \frac{(n-j)!}{k_1!} = \frac{n!}{j!}$. We then obtain $|\mathcal{F}_4| = \sum_{j=0}^n \binom{n}{j} \frac{n!}{j!}$. This sum equals $n!\mathcal{L}_n(-1)$ which in turn is equal to $\mathcal{R}_n(1)$. It follows that $|\mathcal{F}| = \sum_{i=1}^4 |\mathcal{F}_i| = (n+1)^2 + n!\mathcal{L}_n(-1)$ as wanted. \square

4 Algorithms

We shall provide efficient algorithms for the maximization problem mentioned in the introduction: Given $S \subseteq \mathfrak{J}(L)$ find $\prod_{\mathfrak{J}(L)} S$, i.e., the greatest join-endomorphism in the lattice $\mathfrak{J}(L)$ below all the elements of S .

Finding $\prod_{\mathfrak{J}(L)} S$ may not be immediate. E.g., see $\prod_{\mathfrak{J}(L)} S$ in Fig. 1a for a small lattice of four elements and two join-endomorphisms. As already mentioned, a *naive approach* is to compute $\prod_{\mathfrak{J}(L)} S$ by taking $\sigma_S(c) \stackrel{\text{def}}{=} \prod_L \{f(c) \mid f \in S\}$ for each $c \in L$. This does not work since σ_S is not necessarily a join-endomorphism as shown in Fig. 1b.

A *brute force* solution to computing $\prod_{\mathfrak{J}(L)} S$ can be obtained by generating the set $S' = \{g \mid g \in \mathfrak{J}(L) \text{ and } g \sqsubseteq f \text{ for all } f \in S\}$ and taking its join. This approach works since $\prod S' = \prod_{\mathfrak{J}(L)} S$ but as shown in Section 3, the size of $\mathfrak{J}(L)$ can be super-polynomial for distributive lattices and super-exponential in general.

Nevertheless, one can use lattice properties to compute $\prod_{\mathfrak{J}(L)} S$ efficiently. For distributed lattices, we use the inherent compositional nature of $\prod_{\mathfrak{J}(L)} S$. For arbitrary lattices, we present an algorithm that uses the function σ_S in the naive approach to compute $\prod_{\mathfrak{J}(L)} S$ by approximating it from above.

We will give the time complexities in terms of the number of basic binary lattice operations (i.e., meets, joins and subtractions) performed during execution.

4.1 Meet of Join-Endomorphisms in Distributed Lattices

Here we shall illustrate some pleasant compositionality properties of the infima of join-endomorphisms that can be used for computing the join-endomorphism $\prod_{\mathfrak{J}(L)} S$ in a finite distributed lattice L . In what follows we assume $n = |L|$ and $m = |S|$.

We use X^J to denote the set of tuples $(x_j)_{j \in J}$ of elements $x_j \in X$ for each $j \in J$.

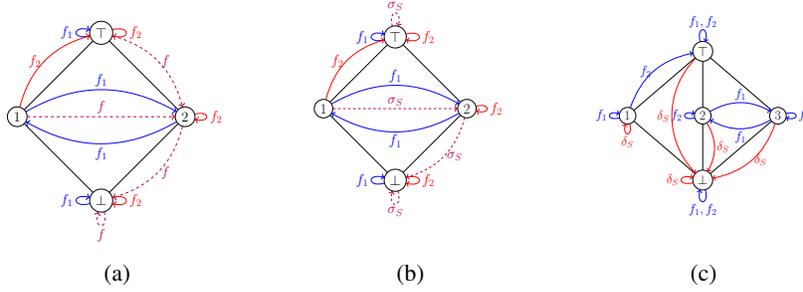


Fig. 1: $S = \{f_1, f_2\} \subseteq \mathfrak{J}(L)$. (a) $f = \prod_{\mathfrak{J}(L)} S$. (b) $\sigma_S(c) \stackrel{\text{def}}{=} f_1(c) \sqcap f_2(c)$ is not a join-endomorphism of \mathbf{M}_2 : $\sigma_S(1 \sqcup 2) \neq \sigma_S(1) \sqcup \sigma_S(2)$. (c) δ_S in Lemma 1 is not a join-endomorphism of the non-distributive lattice \mathbf{M}_3 : $\delta_S(1) \sqcup \delta_S(2) = 1 \neq \perp = \delta_S(1 \sqcup 2)$.

Lemma 1. Let L be a finite distributive lattice and $S = \{f_i\}_{i \in I} \subseteq \mathfrak{J}(L)$. Then $\prod_{\mathfrak{J}(L)} S = \delta_S$ where $\delta_S(c) \stackrel{\text{def}}{=} \prod_L \{ \bigsqcup_{i \in I} f_i(a_i) \mid (a_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \}$.

The above lemma basically says that $(\prod_{\mathfrak{J}(L)} S)(c)$ is the greatest element in L below all possible applications of the functions in S to elements whose join is greater or equal to c . The proof that $\delta_S \sqsupseteq_{\mathfrak{J}} \prod_{\mathfrak{J}(L)} S$ uses the fact that join-endomorphisms preserve joins. The proof that $\delta_S \sqsubseteq_{\mathfrak{J}} \prod_{\mathfrak{J}(L)} S$ proceeds by showing that δ_S is a lower bound in $\mathfrak{J}(L)$ of S . Distributivity of the lattice L is crucial for this direction. In fact without it $\prod_{\mathfrak{J}(L)} S = \delta_S$ does not necessarily hold as shown by the following counter-example.

Example 1. Consider the non-distributive lattice \mathbf{M}_3 and $S = \{f_1, f_2\}$ defined as in Fig.1c. We obtain $\delta_S(1 \sqcup 2) = \delta_S(\top) = \perp$ and $\delta_S(1) \sqcup \delta_S(2) = 1 \sqcup \perp = 1$. Then, $\delta_S(1 \sqcup 2) \neq \delta_S(1) \sqcup \delta_S(2)$, i.e., δ_S is not a join-endomorphism.

Algorithm A₁. One could use Lemma 1 directly in the obvious way to provide an algorithm, call it A_1 , for $\prod_{\mathfrak{J}(L)} S$ by computing δ_S : i.e., computing the meet of elements of the form $\bigsqcup_{i \in I} f_i(a_i)$ for every tuple $(a_i)_{i \in I}$ such that $\bigsqcup_{i \in I} a_i \sqsupseteq c$. For each $c \in L$, $\delta_S(c)$ checks n^m tuples $(a_i)_{i \in I}$, each one with a cost in $O(m)$. Thus A_1 can compute $\prod_{\mathfrak{J}(L)} S$ by performing $O(n \times n^m \times m) = O(mn^{m+1})$ binary lattice operations.

Nevertheless, we can use Lemma 1 to provide a recursive characterization of $\prod_{\mathfrak{J}(L)} S$ that can be used in a divide-and-conquer algorithm with lower time complexity.

Proposition 3. Let L be a finite distributive lattice and $S = S_1 \cup S_2 \subseteq \mathfrak{J}(L)$. Then $(\prod_{\mathfrak{J}(L)} S)(c) = \prod_L \{ (\prod_{\mathfrak{J}(L)} S_1)(a) \sqcup (\prod_{\mathfrak{J}(L)} S_2)(b) \mid a, b \in L \text{ and } a \sqcup b \sqsupseteq c \}$.

The above proposition bear witness to the compositional nature of $\prod_{\mathfrak{J}(L)} S$ in distributed lattices. It can be proven by replacing $(\prod_{\mathfrak{J}(L)} S_1)(a)$ and $(\prod_{\mathfrak{J}(L)} S_2)(b)$ by $\delta_{S_1}(a)$ and $\delta_{S_2}(b)$ using Lemma 1 and the distributivity in L of joins over meets.

Algorithm A₂. We can use Prop.3 to compute $\prod_{\mathfrak{J}(L)} S$ recursively: Take any partition $\{S_1, S_2\}$ of S such that the absolute value of $|S_1| - |S_2|$ is at most 1. Then compute the meet of all $(\prod_{\mathfrak{J}(L)} S_1)(a) \sqcup (\prod_{\mathfrak{J}(L)} S_2)(b)$ for every a, b such that $a \sqcup b \sqsupseteq c$. Then given

$c \in L$, the time complexity for computing $(\prod_{\mathfrak{J}(L)} S)(c)$ can be obtained as the solution of the equation $T(m) = n^2(1 + 2T(m/2))$ and $T(1) = 1$ which is in $O(mn^{2 \log_2 m})$. Therefore, $\prod_{\mathfrak{J}(L)} S$ can be computed in $O(mn^{1+2 \log_2 m})$.

Certainly, the time complexity of Algorithm A_2 is better than that of Algorithm A_1 . However, we will show that one can compute $\prod_{\mathfrak{J}(L)} S$ in a much lower complexity order.

4.2 Using Subtraction and Downsets to characterize $\prod_{\mathfrak{J}(L)} S$

In what follows we show the main result of this section: $\prod_{\mathfrak{J}(L)} S$ can be computed in $O(mn^{\log_2 3})$ for powerset lattices, and in $O(mn^2)$ for lattices of sets. To achieve this we use the subtraction operator from co-Heyting algebras and the notion of down set.

Subtraction Operator. Notice that in Prop.3 we are considering all pairs $a, b \in L$ such that $a \sqcup b \sqsupseteq c$. However, because of the monotonicity of join-endomorphisms, it suffices to take, for each $a \in L$, just the least b such that $a \sqcup b \sqsupseteq c$. In finite distributed lattices, and more generally in co-Heyting algebras [5], the subtraction operator $c \setminus a$ gives us exactly such a least element. The subtraction operator is uniquely determined by the property (Galois connection) $b \sqsupseteq c \setminus a$ iff $a \sqcup b \sqsupseteq c$ for all $a, b, c \in L$.

Down-sets. Besides using just $c \setminus a$ instead of all b 's such that $a \sqcup b \sqsupseteq c$, we can use a further simplification: Rather than including every $a \in L$, we only need to consider every a in the down-set of c . Recall that the down-set of c is defined as $\downarrow c = \{e \in L \mid e \sqsubseteq c\}$. This additional simplification is justified using properties of distributive lattices to show that for any $a' \in L$, such that $a' \not\sqsubseteq c$, there exists $a \sqsubseteq c$ such that $(\prod_{\mathfrak{J}(L)} S_1)(a) \sqcup (\prod_{\mathfrak{J}(L)} S_2)(c \setminus a) \sqsubseteq (\prod_{\mathfrak{J}(L)} S_1)(a') \sqcup (\prod_{\mathfrak{J}(L)} S_2)(c \setminus a')$.

The above observations lead us to the following theorem.

Theorem 5. *Let L be a finite distributive lattice and $S = S_1 \cup S_2 \subseteq \mathfrak{J}(L)$. Then $(\prod_{\mathfrak{J}(L)} S)(c) = \prod_L \{(\prod_{\mathfrak{J}(L)} S_1)(a) \sqcup (\prod_{\mathfrak{J}(L)} S_2)(c \setminus a) \mid a \in \downarrow c\}$.*

4.3 Algorithms for Distributed Lattices

We first describe the algorithm DMEETAPP that computes the value $(\prod_{\mathfrak{J}(L)} S)(c)$. We then describe the algorithm DMEET that computes the function $\prod_{\mathfrak{J}(L)} S$ by calling DMEETAPP in a particular order to avoid repeating computations. To specify the calling order we need the following definition.

Definition 2. *A binary partition tree (bpt) of a finite set $S \neq \emptyset$ is a binary tree such that (a) its root is S , (b) if $|S| = 1$ then its root is a leaf, and (c) if $|S| > 1$ it has a left and a right subtree, themselves bpts of S_1 and S_2 resp., for a partition $\{S_1, S_2\}$ of S .*

Let Δ be a bpt of S . We use $\Delta(S')$ for the subtree of Δ rooted at $S' \subseteq S$, if it exists. Clearly, $\Delta = \Delta(S)$. We use the triple $\langle S, \Delta_1, \Delta_2 \rangle$ for the bpt of S with Δ_1 and Δ_2 as its left and right subtrees.

The following proposition is an immediate consequence of the previous definition.

Proposition 4. *The size (number of nodes) of any bpt of S is $2m - 1$ where $m = |S|$.*

DMEETAPP(Δ, c). Let $\Delta = \langle S, \Delta_1, \Delta_2 \rangle$ be a bpt of $S \subseteq \mathfrak{J}(L)$ where L is a distributive lattice. The recursive program **DMEETAPP**(Δ, c) defined in Algorithm 1 computes $(\prod_{\mathfrak{J}(L)} S)(c)$. It uses a global lookup table T for storing the results of calls to **DMEETAPP**. Initially each entry of T stores a `null` value not included in L . Since S is the union of the roots of Δ_1 and Δ_2 , the correctness of **DMEETAPP**(Δ, c) follows from Thm.5. Termination follows from the fact that L is finite and the bpts Δ_1 and Δ_2 in the recursive calls are strictly smaller than Δ .

Algorithm 1 **DMEETAPP**(Δ, c) returns $(\prod_{\mathfrak{J}(L)} S)(c)$ where Δ is a bpt of $S \subseteq \mathfrak{J}(L)$ and L is a finite distributive lattice. The global variable T is used as a lookup table.

```

1: procedure DMEETAPP( $\Delta, c$ )                                ▷  $\Delta = \langle S, \Delta_1, \Delta_2 \rangle$  is a bpt of  $S$ 
2:   if IsNull( $T[S, c]$ ) then                                ▷ Test if  $T[S, c]$  does not store yet a value from  $L$ .
3:     if  $S = \{f\}$  then
4:        $T[S, c] \leftarrow f(c)$ 
5:     else
6:        $T[S, c] \leftarrow \prod_L \{ \text{DMEETAPP}(\Delta_1, a) \sqcup \text{DMEETAPP}(\Delta_2, c \setminus a) \mid a \in \downarrow c \}$ .
7:   return  $T[S, c]$ 

```

Computing $\prod_{\mathfrak{J}(L)} S$ for Lattices of Sets. Recall that every finite distributive lattice is isomorphic to a lattice of sets and that any finite boolean algebra is isomorphic to a powerset lattice [2]. We show how to compute $\prod_{\mathfrak{J}(L)} S$ with a worst-case time complexity in $O(mn^2)$ for lattices of sets and in $O(mn^{\log_2 3})$ for powerset lattices.

Let L be a finite lattice of sets and $\Delta = \langle S, \Delta_1, \Delta_2 \rangle$ be a bpt of $S \subseteq \mathfrak{J}(L)$. Let $n = |L|$ and $m = |S|$. Let us consider an execution of **DMEETAPP**(Δ, c). From the definition of subtraction it follows that $c \setminus a \in \downarrow c$. Then for each recursive call **DMEETAPP**(Δ', a') performed by an execution of **DMEETAPP**(Δ, c) we have $a' \in \downarrow c$.

The above leads us to the following observation about the order of the number of binary lattice operations (meets, joins, and subtractions) performed by **DMEETAPP**(Δ, c).

Observation 6 *Let $\Delta = \langle S, \Delta_1, \Delta_2 \rangle$ with Δ_1 and Δ_2 rooted at S_1 and S_2 . Assume that $T[S_1, a'], T[S_2, a'] \in L$ for every $a' \in \downarrow c$. Then the number of binary lattice operations performed by **DMEETAPP**(Δ, c) is in $O(|\downarrow c|)$.*

Since each entry of T is initialized with a `null` value not in L , the assumption in Obs.6 implies that for every $a' \in \downarrow c$ the values of **DMEETAPP**(Δ_1, a') and **DMEETAPP**(Δ_2, a') have been previously stored in T . Under this condition **DMEETAPP**(Δ, c) performs at most $|\downarrow c|$ binary joins, $|\downarrow c|$ subtractions, $|\downarrow c| - 1$ binary meets.

DMEET(L, S). The join-endomorphism $\prod_{\mathfrak{J}(L)} S$ can be computed by the program in Algorithm 2 as follows. The program first initializes each entry of table T with a `null` value. Then, to satisfy the assumption in Obs.6, it traverses a partition-tree Δ of S and the lattice L as follows: It visits each node S' of Δ in *post-order* (i.e., before visiting

Algorithm 2 $\text{DMEET}(L, S)$ finds $\prod_{\mathfrak{J}(L)} S$ for $S \subseteq \mathfrak{J}(L)$ with $|S| \geq 1$. L is a finite lattice of sets and Δ is a partition-tree of S

```

1:  $T(S', a) \leftarrow \text{null}$  ▷ for each  $a \in L$  and each node  $S'$  of  $\Delta$ 
2: for each  $S'$  in a post-order traversal sequence of  $\Delta$  do ▷ Visit each node in post-order
3:   for  $i = 0$  to  $|\top|$  do ▷ Traversing  $L$ 
4:      $\text{DMEETAPP}(\Delta(S'), c_i)$  ▷ for each set  $c_i \in L$  s.t.  $|c_i| = i$ 
5:    $f(c) \leftarrow T[S, c]$  ▷ for each  $c \in L$ 
6: return  $f$ 
    
```

a node it first visits its children). For each subtree $\Delta(S')$ of Δ , it computes $\text{DMEETAPP}(\Delta(S'), c_i)$ for every $c_i \in L$ of size i , with $0 \leq i \leq h$ and h the size of the top element of L . The correctness of $\text{DMEET}(L, S)$ follows from that of $\text{DMEETAPP}(\Delta, c)$.

Complexity for arbitrary lattices of sets. It is easy to see that the above-mentioned traversals of Δ and L ensure that the assumption in Obs.6 is satisfied by each call of the form $\text{DMEETAPP}(\Delta(S'), c_i)$ performed during the execution of $\text{DMEET}(L, S)$. From Prop.4 we know that the number of iterations of the outer **for** is $2m - 1$. Clearly $|\downarrow c|$ is in $O(n)$. Thus for any given S' , we conclude from Obs.6 that the total number of operations from all calls of the form $\text{DMEETAPP}(\Delta(S'), c_i)$, executed in the inner **for**, is in $O(n^2)$. The worst-case time complexity of $\text{DMEET}(L, S)$ is then in $O(mn^2)$.

Complexity for Powerset Lattices. Assume that the lattice of sets L is a powerset lattice of size n . From Prop.4, the initialization (Line 1) takes $O(nm)$. Let h be the cardinality of the top element of L . Since L is a powerset, $n = 2^h$, $|\downarrow c| = 2^i$ where $i = |c|$, and the number of elements in L of size i is $\binom{h}{i}$. Thus for any given S' and i , it follows from Obs.6 that the number of operations from all calls of the form $\text{DMEETAPP}(\Delta(S'), c_i)$ is in the order of $2^i \times \binom{h}{i}$. Then for any given S' , the number of operations from all calls of the form $\text{DMEETAPP}(\Delta(S'), c_i)$, executed by the inner **for**, is in the order of $\sum_{i=0}^h 2^i \times \binom{h}{i} = 3^h$. But $n = 2^h$, then $3^h = 3^{\log_2 n} = n^{\log_2 3}$. Using Prop.4 we conclude that the worst-case time complexity of $\text{DMEET}(L, S)$ is in $O(mn^{\log_2 3})$.

4.4 Algorithm for Arbitrary Lattices

The previous algorithm may fail to produce the $\prod_{\mathfrak{J}(L)} S$ for non-distributive finite lattices. Nonetheless, for any arbitrary finite lattice L , $\prod_{\mathfrak{J}(L)} S$ can be computed by successive approximations, starting with some self-map known to be smaller than each $f \in S$ and greater than $\prod_{\mathfrak{J}(L)} S$. Assume a self-map $\sigma : L \rightarrow L$ such that $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$ and, for all $f \in S$, $\sigma \sqsubseteq f$. A good starting point is $\sigma(u) = \prod\{f(u) \mid f \in S\}$, for all $u \in L$. By definition of \prod , $\sigma(u)$ is the biggest function under all functions in S , hence $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$. The program GMEET in Algorithm 3 computes decreasing upper bounds of $\prod_{\mathfrak{J}(L)} S$ by correcting σ values not conforming to the following *join-endomorphism property*: $\sigma(u) \sqcup \sigma(v) = \sigma(u \sqcup v)$. The correction decreases σ and maintains the invariant $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$, as stated in Thm.7.

Theorem 7. *Let L be a finite lattice, $u, v \in L$, $\sigma : L \rightarrow L$ and $S \subseteq \mathfrak{J}(L)$. Assume $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$ holds, and consider the following updates:*

1. when $\sigma(u) \sqcup \sigma(v) \sqsubset \sigma(u \sqcup v)$, assign $\sigma(u \sqcup v) \leftarrow \sigma(u) \sqcup \sigma(v)$
2. when $\sigma(u) \sqcup \sigma(v) \not\sqsubseteq \sigma(u \sqcup v)$, assign $\sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v)$ and also $\sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v)$

Let σ' be the function resulting after the update. Then, (1) $\sigma' \sqsubset \sigma$ and (2) $\sigma' \sqsupseteq \prod_{\mathfrak{A}(L)} S$.

Algorithm 3 GMEET finds $\sigma = \prod_{\mathfrak{A}(L)} S$

```

1:  $\sigma(u) \leftarrow \prod \{f(u) \mid f \in S\}$  ▷ for all  $u \in L$ 
2: while  $u, v \in L \wedge \sigma(u) \sqcup \sigma(v) \neq \sigma(u \sqcup v)$  do
3:   if  $\sigma(u) \sqcup \sigma(v) \sqsubset \sigma(u \sqcup v)$  then ▷ case (1)
4:      $\sigma(u \sqcup v) \leftarrow \sigma(u) \sqcup \sigma(v)$ 
5:   else ▷ case (2)
6:      $\sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v)$ 
7:      $\sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v)$ 

```

The procedure (see Algo.3) loops through pairs $u, v \in L$ while there is some pair satisfying cases (1) or (2) above for the current σ . When there is, it updates σ as mentioned in Thm.7. At the end of the loop all pairs $u, v \in L$ satisfy the join preservation property. By the invariant mentioned in the theorem, this means $\sigma = \prod_{\mathfrak{A}(L)} S$.

As for the previous algorithms in this paper the worst-time time complexity will be expressed in terms of the binary lattice operations performed during execution. Assume a fixed set S of size m . The complexity of the initialization (Line 1) of GMEET is $O(nm)$ with $n = |L|$. The value of σ for a given $w \in L$ can be updated (decreased) at most n times. Thus, there are at most n^2 updates of σ for all values of L . Finding a $w = u \sqcup v$ where $\sigma(w)$ needs an update because $\sigma(u) \sqcup \sigma(v) \neq \sigma(u \sqcup v)$ (test of the loop, Line 2) takes $O(n^2)$. Hence, the worst time complexity of the loop is in $O(n^4)$.

The program GMEET+ in Algo.4 uses appropriate data structures to reduce significantly the time complexity of the algorithm. Essentially, different sets are used to keep track of properties of (u, v) lattice pairs with respect to the current σ . We have a support (correct) pairs set $\text{Sup}_w = \{(u, v) \mid w = u \sqcup v \wedge \sigma(u) \sqcup \sigma(v) = \sigma(w)\}$. We also have a conflicts set $\text{Con}_w = \{(u, v) \mid w = u \sqcup v \wedge \sigma(u) \sqcup \sigma(v) \sqsubset \sigma(w)\}$ and failures set $\text{Fail}_w = \{(u, v) \mid w = u \sqcup v \wedge \sigma(u) \sqcup \sigma(v) \not\sqsubseteq \sigma(w)\}$.

Algorithm 4 updates σ as mentioned in Thm.7 and so maintains the invariant $\sigma \sqsupseteq \prod_{\mathfrak{A}(L)} S$. An additional invariant is that, for all w , sets $\text{Sup}_w, \text{Con}_w, \text{Fail}_w$ are such $\text{Sup}_w \cup \text{Con}_w \cup \text{Fail}_w = \{(u, v) \mid u \sqcup v = w\}$ and $\text{Sup}_w \cap \text{Con}_w \cap \text{Fail}_w = \emptyset$. When the outer loop finishes sets Con_w and Fail_w are empty (for all w) and thus every (u, v) belongs to $\text{Sup}_{u \sqcup v}$, i.e. the resulting $\sigma = \prod_{\mathfrak{A}(L)} S$.

Auxiliary procedure CHECKSUPPORTS(u) identifies all pairs of the form $(u, x) \in \text{Sup}_{u \sqcup x}$ that may no longer satisfy the join-endomorphism property $\sigma(u) \sqcup \sigma(x) = \sigma(u \sqcup x)$ because of an update to $\sigma(u)$. When this happens, it adds (u, x) to the appropriate Con, or Fail set. The time complexity of the algorithm depends on the set operations computed for each $w \in L$ chosen, either in the *conflicts* Con_w set or in the *failures* Fail_w set. When a w is selected (for some (u, v) s.t. $u \sqcup v = w$) the following

Algorithm 4 GM_{EET}+ finds $\sigma = \prod_{3(L)} S$

```

1:  $\sigma(u) \leftarrow \prod\{f(u) \mid f \in S\}$  ▷ for all  $u \in L$ 
2: Initialize  $\text{Sup}_w, \text{Con}_w, \text{Fail}_w$ , for all  $w$ 
3: while  $w \in L$  s.t.  $(u, v) \in \text{Con}_w$  do ▷ some conflict set not empty
4:    $\text{Con}_w \leftarrow \text{Con}_w \setminus \{(u, v)\}$ 
5:    $\sigma(w) \leftarrow \sigma(u) \sqcup \sigma(v)$ 
6:    $\text{Fail}_w \leftarrow \text{Fail}_w \cup \text{Sup}_w$  ▷ all pairs previously in  $\text{Sup}_w$  are now failures
7:    $\text{Sup}_w \leftarrow \{(u, v)\}$ 
8:   CHECKSUPPORTS( $w$ ) ▷ for  $u \in L$ , verify property  $\text{Sup}_{w \sqcup u}$ 
9:   while  $z \in L$  s.t.  $(x, y) \in \text{Fail}_z$  do ▷ some failures set not empty
10:     $\text{Fail}_z \leftarrow \text{Fail}_z \setminus \{(x, y)\}$ 
11:    if  $\sigma(x) \neq \sigma(x) \sqcap \sigma(z)$  then
12:       $\sigma(x) \leftarrow \sigma(x) \sqcap \sigma(z)$  ▷  $\sigma(x)$  decreases
13:       $\text{Fail}_x \leftarrow \text{Fail}_x \cup \text{Sup}_x$  ▷ all pairs in  $\text{Sup}_x$  are now failures
14:       $\text{Sup}_x \leftarrow \emptyset$ 
15:      CHECKSUPPORTS( $x$ ) ▷ for  $u \in L$ , verify property  $\text{Sup}_{x \sqcup u}$ 
16:      if  $\sigma(y) \neq \sigma(y) \sqcap \sigma(z)$  then
17:         $\sigma(y) \leftarrow \sigma(y) \sqcap \sigma(z)$  ▷  $\sigma(y)$  decreases
18:         $\text{Fail}_y \leftarrow \text{Fail}_y \cup \text{Sup}_y$  ▷ all pairs in  $\text{Sup}_y$  are now failures
19:         $\text{Sup}_y \leftarrow \emptyset$ 
20:        CHECKSUPPORTS( $y$ ) ▷ for  $u \in L$ , verify property  $\text{Sup}_{y \sqcup u}$ 
21:      if  $\sigma(x) \sqcup \sigma(y) = \sigma(z)$  then
22:         $\text{Sup}_z \leftarrow \text{Sup}_z \cup \{(x, y)\}$  ▷  $(x, y)$  is now correct
23:      else
24:         $\text{Con}_z \leftarrow \text{Con}_z \cup \{(x, y)\}$  ▷  $(x, y)$  is now a conflict

```

holds: (1) at least one of $\sigma(w), \sigma(u), \sigma(v)$ is decreased, (2) some fix k number of elements are removed from or added to a set, (3) a union of two *disjoint* sets is computed, and (4) new support sets of w, u or v are calculated.

With an appropriate implementation, operations (1)-(2) take $O(1)$, and also operation (3), since sets are disjoint. Operation (4) clearly takes $O(n)$. In each loop of the (outer or inner) cycles of the algorithm, at least one σ reduction is computed. Furthermore, for each reduction of σ , $O(n)$ operations are performed. The maximum possible number of $\sigma(w)$ reductions, for a given w , is equal to the length d of the longest strictly decreasing chain in the lattice. The total number of possible σ reductions is thus equal to nd . The total number of operations of the algorithm is then $O(n^2d)$. In general, d could be (at most) equal to n , therefore, after initialization, worst case complexity is $O(n^3)$. The initialization (Lines 1-2) takes $O(nm) + O(n^2)$, where $m = |S|$. Worst time complexity is thus $O(mn + n^3)$. For powerset lattices, $d = \log_2 n$, thus worst time complexity in this case is $O(mn + n^2 \log_2 n)$.

4.5 Experimental Results and Small Example

Here we present some experimental results showing the execution time of the proposed algorithms. We also discuss a small example with join-endomorphisms representing

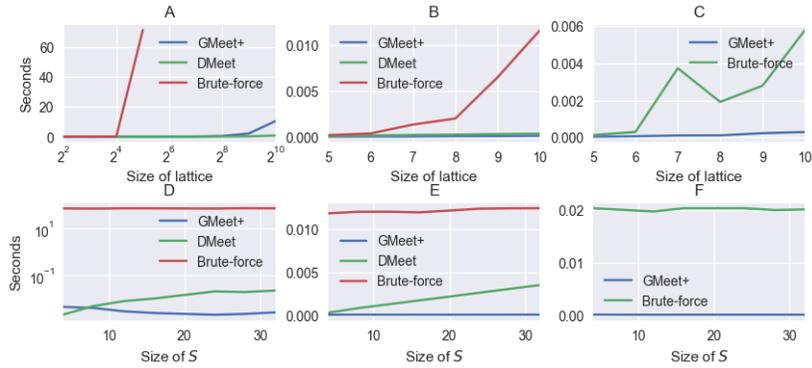


Fig. 2: Average performance time of GMEET+, DMEET and BRUTE-FORCE. Plots A and D use 2^n lattices, B and E distributive lattices, and C and F arbitrary (possibly non-distributive) lattices. Plots A-C have a fixed number of join-endomorphisms and plots D-F have a fixed lattice size.

dilation operators from Mathematical Morphology [1]. We use the algorithms presented above to compute the greatest dilation below a given set of dilations and illustrate its result for a simple image.

Consider Figure 2. In plots 2.A-C, horizontal axis is the size of the lattice. In plots 2.D-F, horizontal axis is the size of S . Curves in images 2.A-C plot, for each algorithm, the average execution time of 100 runs (10 for 2.A) with random sets $S \subseteq \mathfrak{J}(L)$ of size 4. Images 2.D-F, show the mean execution time of each algorithm for 100 runs (10 for 2.D) varying the number of join-endomorphisms ($|S| = 4i, 1 \leq i \leq 8$). The lattice size is fixed: $|L| = 10$ for 2.E and 2.F, and $|L| = 2^5$ for 2.D. For a given lattice L and $S \subseteq \mathfrak{J}(L)$, the brute-force algorithm explores the whole space $\mathfrak{J}(L)$ to find all the join-endomorphism below each element of S and then computes the greatest of them. In particular, the measured spike in plot 2.C corresponds to the random lattice of seven elements with the size of $\mathfrak{J}(L)$ being bigger than in the other experiments in the same figure. In our experiments we observed that for a fixed S , as the size of the lattice increases, DMEET outperforms GMEET+. This is noticeable in lattices 2^n (see 2.A). Similarly, for a fixed lattice, as the size of S increases GMEET+ outperforms DMEET. GMEET+ performance can actually improve with a higher number of join-endomorphisms (see 2.D) since the initial σ is usually smaller in this case.

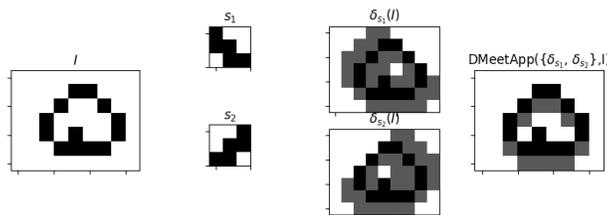
To illustrate some performance gains, Table 2 shows the mean execution time of the algorithms discussed in this paper. We include A_1 and A_2 , the algorithms outlined just after Lemma 1 and Proposition 3.

An MM Example. Mathematical morphology (MM) is a theory, based on topological, lattice-theoretical and geometric concepts, for the analysis of geometric structures. Its algebraic framework comprises [1,11,12], among others, complete lattices together with certain kinds of morphisms, such as *Dilations*, defined as *join-endomorphisms* [11]. Our results give bounds about the number of all dilations over certain specific finite lattices and also efficient algorithms to compute their infima.

Size	A_1	A_2	GMEET	GMEET+	DMEET
16	2.01	0.958	0.00360	0.000603	0.000632
32	64.6	25.3	0.0633	0.00343	0.00181
64	1901	600	0.948	0.0154	0.00542
128	>600	>600	15.4	0.0860	0.0160
256	>600	>600	252	0.361	0.0483
512	>600	>600	>600	2.01	0.166
1024	>600	>600	>600	10.7	0.547

Table 2: Average time in seconds over powerset lattices with $|S| = 4$

A typical application of MM is image processing. A binary image on a digital space $G = \mathbb{Z}^2$ can be expressed as $I \subseteq G$, where I is defined as the set of pixels that are set to black, or activated. In this setting, a dilation δ_{s_i} describes an interaction of an image with a *structuring element* $s_i \subseteq D$, e.g. a pixel is part of the dilated set if, when placing the structuring element anywhere in the image, one of its points hits (overlaps) with the set I . Let L be the powerset lattice for some finite set $D \subseteq G$. It turns out that the dilation $\prod_{\mathfrak{J}(L)} S$ corresponds to the intersection of the structuring elements of the corresponding dilations in S . Fig.3 illustrates $\prod_{\mathfrak{J}(L)} S$ for the two given dilations.

Fig. 3: Dilations $\delta_{s_1}, \delta_{s_2}$, binary image I (on the left). $(\prod_{\mathfrak{J}(L)} \{\delta_{s_1}, \delta_{s_2}\})(I)$ on the right. New elements of the image after each operation in grey.

5 Conclusions and Related Work

We have shown that in the worst-case $\prod_{\mathfrak{J}(L)} S$ can be computed in $O(mn^{\log_2 3})$ binary lattice operations for powerset lattices, $O(mn^2)$ for lattices of sets, and $O(nm + n^3)$ for arbitrary lattices, where $n = |L|$ and $m = |S|$. We illustrated the experimental performance of our algorithms and a small example from mathematical morphology. We have determined the cardinality of the set of join-endomorphisms $\mathfrak{J}(L)$ for significant families of finite lattices. Namely, $n^{\log_2 n}$ for powersets (boolean algebras), central binomial coefficient $\binom{2n}{n}$ for linear orders, and $(n+1)^2 + n! \mathcal{L}_n(-1)$, where $\mathcal{L}_n(x)$ is a Laguerre polynomial, for the lattice \mathbf{M}_n .

The lattice $\mathfrak{J}(L)$ have been studied in [6]. The authors showed that a finite lattice L is distributive iff $\mathfrak{J}(L)$ is distributive. A lower bound of $2^{2n/3}$ for the number of monotonic

self-maps of any finite poset L is given in [3]. Nevertheless to the best of our knowledge, no other authors have studied the problem of determining the size $\mathfrak{J}(L)$ nor algorithms for computing $\prod_{\mathfrak{J}(L)} S$. We believe that these problems are important, as argued in the Introduction, algebraic structures consisting of a lattice and join-endomorphisms are very common in mathematics and computer science. In fact, our interest in this subject arose in the algebraic setting of spatial and epistemic constraint systems [8] where continuous join-endomorphisms, called space functions, represent knowledge and the infima of endomorphisms correspond to distributed knowledge. We showed in [8] that distributed knowledge can be computed in $O(mn^{1+\log_2(m)})$ for distributed lattices and $O(n^4)$ in general. In this paper we have provided much lower complexity orders for computing infima of join-endomorphisms. Furthermore [8] does not provide the exact cardinality of the set of space function of a given lattice.

As future work we plan to explore in detail the applications of our work in mathematical morphology. Furthermore, in the same spirit of [9] we have developed generators of distributed and arbitrary lattices. We observed that for every lattice L generated of size n , $n^{\log_2 n} \leq |\mathfrak{J}(L)| \leq (n+1)^2 + n! \mathcal{L}_n(-1)$ and if L is distributive $n^{\log_2 n} \leq |\mathfrak{J}(L)| \leq \binom{2n}{n}$. This suggest that the lattices we studied are extreme cases for the value $|\mathfrak{J}(L)|$. We plan to establish if this is the case in future work.

References

1. Bloch, I., Heijmans, H., Ronse, C.: Mathematical Morphology, pp. 857–944. Springer Netherlands (2007)
2. Davey, B.A., Priestley, H.A.: Introduction to lattices and order. Cambridge university press, 2nd edn. (2002)
3. Duffus, D., Rodl, V., Sands, B., Woodrow, R.: Enumeration of order preserving maps. Order (1992)
4. Feller, W.: An introduction to probability theory and its applications. Wiley series in probability and mathematical statistics: Probability and mathematical statistics, Wiley (1971)
5. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M., Scott, D.S.: Continuous lattices and domains. Cambridge University Press (2003)
6. Grätzer, G., Schmidt, E.: On the lattice of all join-endomorphisms of a lattice. Proceedings of The American Mathematical Society - PROC AMER MATH SOC (1958)
7. Guzmán, M., Haar, S., Perchy, S., Rueda, C., Valencia, F.: Belief, Knowledge, Lies and Other Utterances in an Algebra for Space and Extrusion. JLAMP (2016)
8. Guzmán, M., Knight, S., Quintero, S., Ramírez, S., Rueda, C., Valencia, F.: Reasoning about distributed knowledge of groups with infinitely many agents. In: CONCUR (2019)
9. Jipsen, P., Lawless, N.: Generating all finite modular lattices of a given size. Algebra universalis (2015)
10. Knight, S., Palamidessi, C., Panangaden, P., Valencia, F.D.: Spatial and Epistemic Modalities in Constraint-Based Process Calculi. In: CONCUR (2012)
11. Ronse, C.: Why mathematical morphology needs complete lattices. Signal Processing (1990)
12. Stell, J.: Why mathematical morphology needs quantaes (2009)

A Proofs

Proof of Theorem 3. We show that $|\mathfrak{J}(\mathbf{M}_n)|$ is super-exponential and can be expressed in terms of Laguerre polynomials: $|\mathfrak{J}(\mathbf{M}_n)| = (n+1)^2 + n!\mathcal{L}_n(-1)$.

Let $\mathcal{F} = \bigcup_{i=1}^4 \mathcal{F}_i$ where the mutually exclusive \mathcal{F}_i 's are defined in Table 1, and $I = \{1, \dots, n\}$. The proof is divided in two parts: (I) $\mathfrak{J}(\mathbf{M}_n) = \mathcal{F}$ and (II) $|\mathcal{F}| = (n+1)^2 + n!\mathcal{L}_n(-1)$.

Part (I) For $\mathcal{F} \subseteq \mathfrak{J}(\mathbf{M}_n)$, it is easy to verify that each $f \in \mathcal{F}$ is a join-endomorphism.

For $\mathfrak{J}(\mathbf{M}_n) \subseteq \mathcal{F}$ we show that for any function f from \mathbf{M}_n to \mathbf{M}_n if $f \notin \mathcal{F}$, then $f \notin \mathfrak{J}(\mathbf{M}_n)$. Immediately, if $f(\perp) \neq \perp$ then $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Suppose $f(\perp) = \perp$. Let J, K, H be disjoint possibly empty sets s.t. $I = J \cup K \cup H$ and let $j = |J|$, $k = |K|$ and $h = |H|$. The sets J, K, H represent the elements of I mapped by f to \top , to elements of I , and to \perp , resp. More precisely, $\text{Img}(f \upharpoonright_J) = \{\top\}$, $\text{Img}(f \upharpoonright_K) \subseteq I$ and $\text{Img}(f \upharpoonright_H) = \{\perp\}$. Furthermore, for every f either (1) $f(\top) = \perp$, (2) $f(\top) \in I$ or (3) $f(\top) = \top$. For each case we show that $f \notin \mathfrak{J}(\mathbf{M}_n)$.

1. $f(\top) = \perp$. Since $f \notin \mathcal{F}_1$ there is an $e \in I$ s.t. $f(e) \neq \perp$. We have $e \sqsubseteq \top$ but $f(e) \not\sqsubseteq f(\top)$. Then f is not monotonic. From Prop. 1 we conclude $f \notin \mathfrak{J}(\mathbf{M}_n)$.
2. $f(\top) \in I$. Let K_1, K_2 be disjoint possibly empty sets such that $K_1 \cup K_2 = K$, $\text{Img}(f \upharpoonright_{K_1}) = \{f(\top)\}$ and $\text{Img}(f \upharpoonright_{K_2}) \neq \{f(\top)\}$. Notice that if $j > 0$ or $|K_2| > 0$, f is non-monotonic and then $f \notin \mathfrak{J}(\mathbf{M}_n)$.

We then must have $j = 0$ and $K_2 = \emptyset$. Since $\text{Img}(f \upharpoonright_K) = \{f(\top)\}$ and $f \notin \mathcal{F}_2$ then $h > 1$. Therefore there must be $e_1, e_2 \in H$ such that $f(e_1) = f(e_2) = \perp$. This implies $f(e_1 \sqcup e_2) = f(\top) \neq \perp = f(e_1) \sqcup f(e_2)$, therefore $f \notin \mathfrak{J}(\mathbf{M}_n)$.

3. $f(\top) = \top$.
 - 3.1. Suppose $k = 0$. Notice that $f \notin \mathcal{F}_3$ and $f \notin \mathcal{F}_4$ hence $h \neq 1$ and $h \neq 0$. Thus $h > 1$ implies that there are at least two $e_1, e_2 \in H$ s.t. $f(e_1) = f(e_2) = \perp$. But then $f(e_1 \sqcup e_2) = f(\top) = \top \neq \perp = f(e_1) \sqcup f(e_2)$, hence $f \notin \mathfrak{J}(\mathbf{M}_n)$.
 - 3.2. Suppose $k > 0$. Assume $h = 0$. Let K_1, K_2 be disjoint possibly empty sets such that $K_1 \cup K_2 = K$, $\text{Img}(f \upharpoonright_{K_1}) \subseteq J$ and $\text{Img}(f \upharpoonright_{K_2}) \subseteq I \setminus J$. Notice that f is a \perp and \top preserving function and satisfies conditions (a) and (c) of \mathcal{F}_4 but $f \notin \mathcal{F}_4$, then f must violate condition (b). Thus $f \upharpoonright_{K_1}$ or $f \upharpoonright_{K_2}$ is not injective. Let us assume that $f \upharpoonright_{K_1}$ is not injective (the case when $f \upharpoonright_{K_2}$ is not injective is analogous). Then there are $a, b \in K_1$ and $c \in J$, such that $a \neq b$ but $f(a) = f(b) = c$. By definition $\top \notin J$, then $f(a) \sqcup f(b) = c \neq \top = f(a \sqcup b)$. Consequently, $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Assume $h > 0$. There must be $e_1, e_2, e_3 \in I$ such that $f(e_1) = \perp$ and $f(e_2) = e_3$. Notice $e_1 \sqcup e_2 = \top$, we then have $f(e_1) \sqcup f(e_2) = e_3 \neq \top = f(e_1 \sqcup e_2)$. Therefore, $f \notin \mathfrak{J}(\mathbf{M}_n)$.

Part (II) We prove that $|\mathcal{F}| = \sum_{i=1}^4 |\mathcal{F}_i| = (n+1)^2 + n!\mathcal{L}_n(-1)$. Recall that $n = |I|$.

1. $|\mathcal{F}_1| = 1$. There is only one function mapping every element in \mathbf{M}_n to \perp .

2. $|\mathcal{F}_2| = n^2 + n$. Since \top is mapped to an element of I , there are n possibilities to choose such element. If there is an element of I mapped to \perp , for each one of the previous n options there are also n possibilities to choose an element of I to be mapped to \perp . Then, in this case there are n^2 functions. If no element of I is mapped to \perp , then there are n additional functions.
3. $|\mathcal{F}_3| = n$. One of the elements of I is mapped to \perp . All the other elements of I are mapped to \top . Then, there are n functions that can be defined in \mathcal{F}_3 .
4. $|\mathcal{F}_4| = \mathcal{R}_n(1)$. Let $f \in \mathcal{F}_4$ and let J, K_1, K_2 be disjoint possibly empty sets such that $I = J \cup K_1 \cup K_2$, $\text{Img}(f \upharpoonright_J) = \{\top\}$, $\text{Img}(f \upharpoonright_{K_1}) \subseteq J$ and $\text{Img}(f \upharpoonright_{K_2}) \subseteq I \setminus J$, where $f \upharpoonright_{K_1}$ and $f \upharpoonright_{K_2}$ are injective functions. We shall call $j = |J|$ and $k_1 = |K_1|$. For each of the $\binom{n}{j}$ possibilities for J , the elements of K_1 are to be mapped to them by the injective function $f \upharpoonright_{K_1}$. This is possible only if $k_1 \leq j$, so we take $k_1 \leq \min(j, n - j)$. The number of choices of K_1 is $\binom{n-j}{k_1}$ and the number of choices among J that can be targets of those is $\binom{j}{k_1}$. Each of these can be mapped from any permutation of K_1 , so we have $\binom{j}{k_1} k_1! = \frac{j!}{(j-k_1)!}$ possibilities to choose K_1 and map its elements by $f \upharpoonright_{K_1}$ to elements of J . Similarly, the number of possibilities to choose K_2 and map its elements by $f \upharpoonright_{K_2}$ to elements of $I \setminus J$ is $\binom{n-j-k_1}{n-j-k_1} (n-j-k_1)! = \frac{(n-j)!}{k_1!}$. Therefore, $|\mathcal{F}_4| = \sum_{j=0}^n \binom{n}{j} \sum_{k_1=0}^{\min(j, n-j)} \binom{n-j}{k_1} \frac{j!}{(j-k_1)!} \frac{(n-j)!}{k_1!}$. We can simplify the inner sum: $\sum_{k_1=0}^{\min(j, n-j)} \binom{n-j}{k_1} \frac{j!}{(j-k_1)!} \frac{(n-j)!}{k_1!} = \frac{n!}{j!}$. We then obtain $|\mathcal{F}_4| = \sum_{j=0}^n \binom{n}{j} \frac{n!}{j!}$. This sum equals $n! \mathcal{L}_n(-1)$ which in turn is equal to $\mathcal{R}_n(1)$.

It follows that $|\mathcal{F}| = \sum_{i=1}^4 |\mathcal{F}_i| = (n+1)^2 + n! \mathcal{L}_n(-1)$ as wanted. \square

A.1 Proof of Lemma 1

Let L be a finite distributive lattice and $S = \{f_i\}_{i \in I} \subseteq \mathfrak{J}(L)$. Then $\prod_{\mathfrak{J}(L)} S = \delta_S$ where $\delta_S(c) \stackrel{\text{def}}{=} \prod_L \{ \bigsqcup_{i \in I} f_i(a_i) \mid (a_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \}$.

Proof. Recall that $\prod_{\mathfrak{J}(L)} S = \max\{h \in \mathfrak{J}(L) \mid h \sqsubseteq_{\mathfrak{J}} g \text{ for all } g \in S\}$ and let us define $\Gamma = \{ \bigsqcup_{i \in I} f_i(a_i) \mid (a_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \}$. We prove (1) $\prod_{\mathfrak{J}(L)} S \sqsubseteq_{\mathfrak{J}} \delta_S$ and (2) $\delta_S \sqsubseteq_{\mathfrak{J}} \prod_{\mathfrak{J}(L)} S$.

1. $\delta_S \sqsubseteq_{\mathfrak{J}} \prod_{\mathfrak{J}(L)} S$.

We prove (a) $\delta_S \in \mathfrak{J}(L)$ and (b) $\delta_S \sqsubseteq_{\mathfrak{J}} f_i$ for every $f_i \in S$.

- (a) Prove that $\delta_S \sqsubseteq_{\mathfrak{J}} f_i$, for every $f_i \in S$.

Let $c \in L$. From definition of δ_S , for every $i \in I$, the element $f_i(c) = f_i(c) \sqcup \bigsqcup_{j \in I \setminus \{i\}} f_j(\perp) \in \Gamma$. Then for every $c \in L$, $\delta_S(c) \sqsubseteq f_i(c)$. Therefore for every $f_i \in S$, $\delta_S \sqsubseteq_{\mathfrak{J}} f_i$.

- (b) $\delta_S \in \mathfrak{J}(L)$.

We show that for any $H \subseteq L$, $\delta_S(\bigsqcup H) = \bigsqcup \{ \delta_S(e) \mid \text{for every } e \in H \}$. Since H is finite, it suffices to show that our claim holds for $H = \emptyset$ and $H = \{c, d\}$. Assume $H = \emptyset$. One can verify that $\delta_S(\perp) = \perp$.

Assume $H = \{c, d\}$. Firstly, we prove that δ_S is monotonic. Suppose $c \sqsupseteq d$. For any $(a_i)_{i \in I} \in L^I$ such that $\bigsqcup_{i \in I} a_i \sqsupseteq c$, we have $\bigsqcup_{i \in I} a_i \sqsupseteq d$. Therefore,

$\{\bigsqcup_{i \in I} f_i(a_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq c\} \subseteq \{\bigsqcup_{i \in I} f_i(a_i) \mid \bigsqcup_{i \in I} a_i \sqsupseteq d\}$ which implies $\delta_S(c) \sqsupseteq \delta_S(d)$.

By monotonicity of δ_S , we know $\delta_S(c \sqcup d) \sqsupseteq \delta_S(c) \sqcup \delta_S(d)$. The other direction follows from the derivation below:

$$\begin{aligned}
 & \delta_S(c) \sqcup \delta_S(d) \\
 = & \langle \text{Definition of } \delta_S(d) \rangle \\
 & \delta_S(c) \sqcup \prod_L \left\{ \bigsqcup_{i \in I} f_i(b_i) \mid (b_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 = & \langle \sqcup \text{ distributes over } \prod \rangle \\
 & \prod_L \left\{ \delta_S(c) \sqcup \bigsqcup_{i \in I} f_i(b_i) \mid (b_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 = & \langle \text{Definition of } \delta_S(c) \rangle \\
 & \prod_L \left\{ \prod_L \left\{ \bigsqcup_{i \in I} f_i(a_i) \mid (a_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \right\} \sqcup \bigsqcup_{i \in I} f_i(b_i) \mid (b_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 = & \langle \sqcup \text{ distributes over } \prod \rangle \\
 & \prod_L \left\{ \prod_L \left\{ \bigsqcup_{i \in I} f_i(a_i) \sqcup \bigsqcup_{i \in I} f_i(b_i) \mid (a_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \right\} \mid (b_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 = & \langle \text{Associativity of } \prod \rangle \\
 & \prod_L \left\{ \bigsqcup_{i \in I} (f_i(a_i) \sqcup f_i(b_i)) \mid (a_i)_{i \in I}, (b_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} a_i \sqsupseteq c \text{ and } \bigsqcup_{i \in I} b_i \sqsupseteq d \right\} \\
 \sqsupseteq & \langle x \sqsupseteq y \text{ and } w \sqsupseteq z \text{ implies } x \sqcup w \sqsupseteq y \sqcup z; c_i = a_i \sqcup b_i; f_i(c_i) = f_i(a_i \sqcup b_i) \rangle \\
 & \prod_L \left\{ \bigsqcup_{i \in I} f_i(c_i) \mid (c_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq c \sqcup d \right\} \\
 = & \langle \text{Definition of } \delta_S(c \sqcup d) \rangle \\
 & \delta_S(c \sqcup d)
 \end{aligned}$$

Thus we conclude $\delta_S \in \mathfrak{J}(L)$.

From items (a) and (b), $\delta_S \sqsubseteq_{\mathfrak{J}} \prod_{\mathfrak{J}(L)} S$ holds.

2. $\prod_{\mathfrak{J}(L)} S \sqsubseteq_{\mathfrak{J}} \delta_S$.

Let $c \in L$ and $(a_i)_{i \in I} \in L^I$ be an arbitrary tuple such that $\bigsqcup_{i \in I} a_i \sqsupseteq c$. Notice that $\bigsqcup_{i \in I} (\prod_{\mathfrak{J}(L)} S)(a_i) \sqsubseteq \bigsqcup_{i \in I} f_i(a_i)$. Since $\prod_{\mathfrak{J}(L)} S$ is a join-endomorphism of L and monotonic, we know that $\bigsqcup_{i \in I} (\prod_{\mathfrak{J}(L)} S)(a_i) = (\prod_{\mathfrak{J}(L)} S)(\bigsqcup_{i \in I} a_i) \sqsupseteq (\prod_{\mathfrak{J}(L)} S)(c)$. Thus $(\prod_{\mathfrak{J}(L)} S)(c) \sqsubseteq \bigsqcup_{i \in I} f_i(a_i)$, i.e., $(\prod_{\mathfrak{J}(L)} S)(c)$ is a lower bound of I . Then for every $c \in L$, $(\prod_{\mathfrak{J}(L)} S)(c) \sqsubseteq \delta_S(c)$. Therefore $\prod_{\mathfrak{J}(L)} S \sqsubseteq_{\mathfrak{J}} \delta_S$.

We conclude $\prod_{\mathfrak{J}(L)} S = \delta_S$.

A.2 Proof of Theorem 4.

We wish to prove that $|A| = \mathcal{R}_n(1)$ where $A = \{f \in \mathfrak{J}(\mathbf{M}_n) \mid f \text{ is non-reducing in } \mathbf{M}_n\}$.

Let $\mathcal{F} = \bigcup_{i=1}^4 \mathcal{F}_i$ where the mutually exclusive \mathcal{F}_i 's are defined in Table 1. In the proof

of Theorem 3 we show that $\mathfrak{J}(\mathbf{M}_n) = \mathcal{F}$ and that $\mathcal{R}_n(1) = |\mathcal{F}_4|$. Notice that every function in \mathcal{F}_4 is non-reducing and every function in $\mathcal{F} \setminus \mathcal{F}_4$ is not non-reducing. Hence $A = \mathcal{F}_4$, thus $|A| = \mathcal{R}_n(1)$.

A.3 Proof of Proposition 3

Let (L, \sqsubseteq) be a finite distributive lattice and $S = \{f_i\}_{i \in I} \subseteq \mathfrak{J}(L)$. Let $S_1, S_2 \subseteq \mathfrak{J}(L)$ be such that $S = S_1 \cup S_2$. Then

$$\left(\prod_{\mathfrak{J}(L)} S\right)(c) = \prod_L \left\{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b) \mid a \sqcup b \sqsupseteq c \right\}.$$

Proof. The proof follows using the fact that $\prod_{\mathfrak{J}(L)} S \in \mathfrak{J}(L)$ and Lemma 1.

Let $a, b, c \in L$ such that $a \sqcup b \sqsubseteq c$. Let $S_1 = \{f_j\}_{j \in J}$ and $S_2 = \{f_k\}_{k \in K}$ such that $I = J \cup K$. From Lemma 1

$$\left(\prod_{\mathfrak{J}(L)} S_1\right)(a) = \prod_L \left\{ \bigsqcup_{j \in J} f_j(a_j) \mid (a_j)_{j \in J} \in L^J \text{ and } \bigsqcup_{j \in J} a_j \sqsupseteq a \right\}$$

and

$$\left(\prod_{\mathfrak{J}(L)} S_2\right)(b) = \prod_L \left\{ \bigsqcup_{k \in K} f_k(b_k) \mid (b_k)_{k \in K} \in L^K \text{ and } \bigsqcup_{k \in K} b_k \sqsupseteq b \right\}.$$

By distributivity of \sqcup over \prod , $\left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b)$ can be written as:

$$\prod_L \left\{ \bigsqcup_{i \in I} f_i(c_i) \mid (c_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq a \sqcup b \right\}$$

where c_i is either a_i , b_i or $a_i \sqcup b_i$.

$$\begin{aligned} & \prod_L \left\{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b) \mid a \sqcup b \sqsupseteq c \right\} \\ &= \langle \text{Construction of } \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b) \rangle \\ & \prod_L \left\{ \prod_L \left\{ \bigsqcup_{i \in I} f_i(c_i) \mid (c_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq a \sqcup b \right\} \mid a \sqcup b \sqsupseteq c \right\} \\ &= \langle \text{Associativity of } \prod \rangle \\ & \prod_L \left\{ \bigsqcup_{i \in I} f_i(c_i) \mid (c_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq a \sqcup b \text{ and } a \sqcup b \sqsupseteq c \right\} \\ &= \langle (\sqsupseteq) \text{Transitivity of } \sqsupseteq; (\sqsubseteq) \text{Associativity of } \prod. \rangle \\ & \prod_L \left\{ \bigsqcup_{i \in I} f_i(c_i) \mid (c_i)_{i \in I} \in L^I \text{ and } \bigsqcup_{i \in I} c_i \sqsupseteq c \right\} \\ &= \langle \text{Lemma 1} \rangle \\ & \left(\prod_{\mathfrak{J}(L)} S\right)(c) \end{aligned}$$

A.4 Proof of Theorem 5

Suppose (L, \sqsubseteq) is a finite distributive lattice. Let $S_1, S_2 \subseteq \mathfrak{J}(L)$ be such that $S = S_1 \cup S_2 \subseteq \mathfrak{J}(L)$. Then

$$\left(\prod_{\mathfrak{J}(L)} S\right)(c) = \prod_L \left\{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a) \mid a \in L \text{ and } a \sqsubseteq c \right\}.$$

Proof. Firstly, we prove that

$$\left(\prod_{\mathfrak{J}(L)} S\right)(c) = \prod_L \left\{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a) \right\}. \quad (1)$$

From its definition $c \setminus a$ represents the least element e such that $a \sqcup e \sqsupseteq c$, i.e., $c \setminus a = \prod \{e \mid a \sqcup e \sqsupseteq c\}$. Take any b such that $a \sqcup b \sqsupseteq c$. Then $b \sqsupseteq c \setminus a$ and since $\prod_{\mathfrak{J}(L)} S_1$ and $\prod_{\mathfrak{J}(L)} S_2$ are monotonic

$$\left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b) \sqsupseteq \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a).$$

From this it follows that

$$\begin{aligned} & \prod_L (C \cup \{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a), \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b) \}) \\ &= \prod_L (C \cup \{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a) \}) \end{aligned}$$

for any $C \subseteq L$. This shows that $\left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(b)$ is redundant since $\left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a)$ is included in the set on the right-hand side of Equation 1.

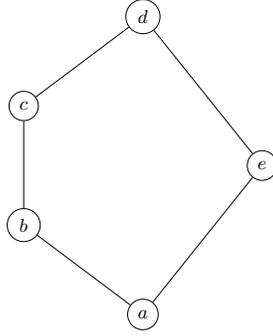
The theorem can be seen as a simplification of Equation 1. Take any $a' \not\sqsubseteq c$. It suffices to find $a \sqsubseteq c$ such that

$$\left(\prod_{\mathfrak{J}(L)} S_1\right)(a') \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a') \sqsupseteq \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a)$$

since then for any $C \subseteq L$

$$\begin{aligned} & \prod_L (C \cup \{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a), \left(\prod_{\mathfrak{J}(L)} S_1\right)(a') \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a') \}) \\ &= \prod_L (C \cup \{ \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a) \}). \end{aligned}$$

Since $a' \not\sqsubseteq c$ either (a) $a' \sqsupset c$ or (b) a' and c are incomparable w.r.t. \sqsubseteq , written $a' \parallel c$. Suppose (a) holds. Then take $a = c$ thus $c \setminus a = \text{true}$. By monotonicity we have $\left(\prod_{\mathfrak{J}(L)} S_1\right)(a') \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a') \sqsupseteq \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a)$ as wanted. Suppose that (b) $a' \parallel c$ holds. Notice that $c \setminus a' \sqsubseteq c$. Suppose that $c \setminus a' = c$. Then we can take $a = \perp$, and thus $c \setminus a = c = c \setminus a'$. By monotonicity we have $\left(\prod_{\mathfrak{J}(L)} S_1\right)(a') \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a') \sqsupseteq \left(\prod_{\mathfrak{J}(L)} S_1\right)(a) \sqcup \left(\prod_{\mathfrak{J}(L)} S_2\right)(c \setminus a)$ as wanted. Suppose $c \setminus a' \sqsubset c$ holds. In this case, which is more interesting, we can build a poset $L = (\{a' \sqcup c, a', c, c \setminus a', a' \sqcap (c \setminus a')\}, \sqsubseteq)$ and verify that L is a non-distributive sub-lattice of (L, \sqsubseteq) , isomorphic to a lattice known as \mathbf{N}_5 (see Fig. 4). But from order theory we know this cannot happen since we assumed (L, \sqsubseteq) to be distributive, and distributive lattices do not have sub-lattices isomorphic to \mathbf{N}_5 ([2]).

Fig. 4: Non-distributive lattice: \mathbf{N}_5 .

A.5 Proof of Theorem 7

Let L be a finite lattice, $u, v \in L$, $\sigma : L \rightarrow L$ and $S \subseteq \mathfrak{J}(L)$. Assume $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$ holds, and consider the following updates:

1. when $\sigma(u) \sqcup \sigma(v) \sqsubseteq \sigma(u \sqcup v)$, assign $\sigma(u \sqcup v) \leftarrow \sigma(u) \sqcup \sigma(v)$
2. when $\sigma(u) \sqcup \sigma(v) \not\sqsubseteq \sigma(u \sqcup v)$, assign $\sigma(u) \leftarrow \sigma(u) \sqcap \sigma(u \sqcup v)$ and also $\sigma(v) \leftarrow \sigma(v) \sqcap \sigma(u \sqcup v)$

Let σ' be the function resulting after the update. Then, (1) $\sigma' \sqsubseteq \sigma$ and (2) $\sigma' \sqsupseteq \prod_{\mathfrak{J}(L)} S$

Proof. For update (1):

given the condition, the assignment obviously decreases $\sigma(u \sqcup v)$, so $\sigma' \sqsubseteq \sigma$. For the invariant, since $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$, then, $\sigma(u) \sqsupseteq (\prod_{\mathfrak{J}(L)} S)(u)$ and $\sigma(v) \sqsupseteq (\prod_{\mathfrak{J}(L)} S)(v)$, and, therefore, $\sigma'(u \sqcup v) = \sigma(u) \sqcup \sigma(v) \sqsupseteq (\prod_{\mathfrak{J}(L)} S)(u) \sqcup (\prod_{\mathfrak{J}(L)} S)(v) = (\prod_{\mathfrak{J}(L)} S)(u \sqcup v)$.

For update (2),

the assignments either decrease $\sigma(u)$ or $\sigma(v)$ (or both). To see why, assume the opposite, $\sigma(u) = \sigma(u) \sqcap \sigma(u \sqcup v) \rightarrow \sigma(u) \sqsubseteq \sigma(u \sqcup v)$, and also $\sigma(v) = \sigma(v) \sqcap \sigma(u \sqcup v) \rightarrow \sigma(v) \sqsubseteq \sigma(u \sqcup v)$. Therefore, $\sigma(u) \sqcup \sigma(v) \sqsubseteq \sigma(u \sqcup v)$, contradicting the condition for update 2. Assignments in update 2 also preserve the invariant $\sigma \sqsupseteq \prod_{\mathfrak{J}(L)} S$.

assume $\sigma(u) \sqcap \sigma(u \sqcup v) \sqsubseteq \sigma(u)$ (otherwise the invariant holds trivially). By the invariant hypothesis for σ before the assignment, we have that $\sigma(u) \sqsupseteq (\prod_{\mathfrak{J}(L)} S)(u)$ and $\sigma(u \sqcup v) \sqsupseteq (\prod_{\mathfrak{J}(L)} S)(u \sqcup v)$. Therefore,

$$\begin{aligned}
 \sigma(u) \sqcap \sigma(u \sqcup v) &\sqsupseteq (\prod_{\mathfrak{J}(L)} S)(u) \sqcap (\prod_{\mathfrak{J}(L)} S)(u \sqcup v) \\
 &= (\prod_{\mathfrak{J}(L)} S)(u) \sqcap ((\prod_{\mathfrak{J}(L)} S)(u) \sqcup (\prod_{\mathfrak{J}(L)} S)(v)) \\
 &= (\prod_{\mathfrak{J}(L)} S)(u) \sqcup ((\prod_{\mathfrak{J}(L)} S)(u) \sqcap (\prod_{\mathfrak{J}(L)} S)(v)) \\
 &= (\prod_{\mathfrak{J}(L)} S)(u)
 \end{aligned}$$

The proof for $\sigma'(v)$ is analogous.