Counting and Computing Join-Endomorphisms in Lattices

Santiago Quintero, Sergio Ramirez, Camilo Rueda, Frank Valencia

Counting and Computing Join-Endomorphisms in Lattices *

Santiago Quintero[2], Sergio Ramirez[1], Camilo Rueda[1], Frank Valencia[1,3]

[1] Pontificia Universidad Javeriana Cali
[2] LIX, École Polytechnique de Paris
[3] 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 $\mathcal{E}(L)$ of all join-endomorphisms of a given finite lattice L. In particular, we show that when L is $\mathbf{M}_n$, the discrete order of n elements extended with top and bottom, $|\mathcal{E}(L)| = n!\mathcal{L}_n(-1) + (n+1)^2$ where $\mathcal{L}_n(x)$ is the Laguerre polynomial of degree n. We also study the following problem: Given a lattice L of size n and a set $S \subseteq \mathcal{E}(L)$ of size m, find the greatest lower bound $\bigsqcap_{\mathcal{E}(L)} S$. The join-endomorphism $\bigsqcap_{\mathcal{E}(L)} S$ 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 $O(n + m \log n)$ 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.

Keywords: join-endomorphisms · lattice cardinality · lattice algorithms.

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 [12]. *Mathematical morphology* (MM), a well-established theory for the analysis and processing of geometrical structures, is founded upon lattice theory [2,14]. 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. A *join-endomorphism* is a function from a lattice to itself that preserves finite joins. 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

* This work has been partially supported by the ECOS-NORD project FACTS (C19M03)

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 $\mathcal{E}(L)$ of all join-endomorphisms of a finite lattice L. The set $\mathcal{E}(L)$ 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 $S \subseteq \mathcal{E}(L)$ of size m, find in $\mathcal{E}(L)$ the greatest lower bound of S, i.e.,* $\bigsqcap_{\mathcal{E}(L)} S$. Simply taking $\sigma : L \to L$ with $\sigma(e) \stackrel{\text{def}}{=} \bigsqcap_L \{f(e) \mid f \in S\}$ does not solve the problem as σ may not be a join-endomorphism. Furthermore, since $\mathcal{E}(L)$ can be seen as the search space, we also consider the problem of determining its cardinality. Our main results are the following.

This paper. We present characterizations of the exact cardinality of $\mathcal{E}(L)$ for some fundamental lattices. Our contribution is to establish the cardinality of $\mathcal{E}(L)$ for the stereotypical non-distributive lattice $L = \mathbf{M}_n$. We show that $|\mathcal{E}(\mathbf{M}_n)|$ equals $r_0^n + \ldots + r_n^n + r_1^{n+1} = n!\mathcal{L}_n(-1) + (n+1)^2$ where r_k^m is the 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. We also present cardinality results for powerset and linear lattices that are part of the lattice theory folklore: The number of join-endomorphisms is $n^{\log_2 n}$ for powerset lattices of size n and $\binom{2n}{n}$ for linear lattices of size $n+1$. Furthermore, we provide algorithms that, given a lattice L of size n and a set $S \subseteq \mathcal{E}(L)$ of size m, compute $\bigsqcap_{\mathcal{E}(L)} S$. Our contribution is to show that $\bigsqcap_{\mathcal{E}(L)} S$ can be computed with worst-case time complexity in $O(n + m\log n)$ for powerset lattices, $O(mn^2)$ for lattices of sets, and $O(nm + n^3)$ for arbitrary lattices.

Due to space restrictions we only include the main proofs. The missing proofs can be found in the technical report of this paper [13].

2 Background: Join-Endomorphisms and Their Space

We presuppose basic knowledge of order theory [3] and use the following notions. Let $(L, \sqsubseteq)$ be a partially ordered 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, $\bigsqcap_L S$ 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 $\bigsqcap_L$ when no confusion

arises. As usual, if $S = \{c, d\}$, $c \sqcup d$ and $c \sqcap d$ represent $\bigsqcup S$ and $\bigsqcap S$, respectively. If L has a greatest element (top) $\top$, and a least element (bottom) $\bot$, we have $\bigsqcup \emptyset = \bot$ and $\bigsqcap \emptyset = \top$. 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 nonempty 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 $f : L \to L$. A self-map f is *monotonic* if $a \sqsubseteq b$ implies $f(a) \sqsubseteq f(b)$. We say that 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 $\{1, \ldots, n\}$ with the linear order $x \sqsubseteq y$ iff $x \leq y$. The poset $\bar{\mathbf{n}}$ is the set $\{1, \ldots, n\}$ with the discrete order $x \sqsubseteq y$ iff $x = y$. Given a poset L, we use $L_\bot$ 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 $(\bar{\mathbf{n}}_\bot)^\top$. 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 a 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 $\mathcal{E}(L)$ as the set of all join-endomorphisms of L. Furthermore, given $f, g \in \mathcal{E}(L)$, define $f \sqsubseteq_\mathcal{E} g$ 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. $f \in \mathcal{E}(L)$ iff $f(\bot) = \bot$ 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 $S \subseteq \mathcal{E}(L)$, where L is a finite lattice, we are interested in finding the greatest join-endomorphism in $\mathcal{E}(L)$ below the elements of S, i.e., $\bigsqcap_{\mathcal{E}(L)} S$. Since every finite lattice is also a complete lattice, the existence of $\bigsqcap_{\mathcal{E}(L)} S$ is guaranteed by the following proposition.

Proposition 2 ([6]). *If $(L, \sqsubseteq)$ is a complete lattice, $(\mathcal{E}(L), \sqsubseteq_\mathcal{E})$ is a complete lattice.*

In the following sections we study the cardinality of $\mathcal{E}(L)$ for some fundamental lattices and provide efficient algorithms to compute $\bigsqcap_{\mathcal{E}(L)} S$.

3 The Size of the Function Space

The main result of this section is Theorem 1. It states the size of $\mathcal{E}(\mathbf{M}_n)$. Propositions 3 and 4 state, respectively, the size of $\mathcal{E}(L)$ for the cases when L is a

powerset lattice and when L is a total order. These propositions follow from simple observations and they are part of the lattice theory folklore [1,10,16]. We include our original proofs of these propositions in the technical report of this paper [13].

3.1 Distributive Lattices

We begin with lattices isomorphic to $\mathbf{2}^n$. They include *finite boolean algebras* and *powerset* lattices [3]. The size of these lattices are easy to infer from the observation that the join-preserving functions on them are determined by their action on the lattices' atoms.

Proposition 3. *Suppose that $m \geq 0$. Let L be any lattice isomorphic to the product lattice $\mathbf{2}^m$. Then $|\mathcal{E}(L)| = n^{\log_2 n}$ 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 allow for an exponential number of join-endomorphisms given by the *central binomial coefficient*. The following proposition is also easy to prove from the observation that the join-endomorphisms over a linear order are also monotonic functions. In fact, this result appears in [1] and it is well-known among the RAMICS community [10,16].

Proposition 4. *Suppose that $n \geq 0$. Let L be any lattice isomorphic to the linear order lattice $\mathbf{n}_\perp$. Then $|\mathcal{E}(L)| = \binom{2n}{n}$.*

It is easy to prove that $\frac{4^n}{2\sqrt{n}} \leq \binom{2n}{n} \leq 4^n$ for $n \geq 1$. Together with Prop.4, this gives us explicit exponential lower and upper bounds for $|\mathcal{E}(L)|$ when L is a linear lattice.

3.2 Non-distributive Case

The number of join-endomorphisms for some non-distributive lattices of a given size can be much bigger than that for those distributive lattices of the same size in the previous section. We will characterize this number for an archetypal non-distributive lattice in terms of Laguerre (and rook) polynomials.

Laguerre polynomials are solutions to Laguerre's second-order linear differential equation $xy'' + (1 - x)y' + ny = 0$ where y' and y'' are the first and second derivatives of an unknown function y of the variable x, and n is a non-negative integer. 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 $\mathcal{E}(\mathbf{M}_n)$ can be succinctly expressed as follows.

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

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 instance, $r(0,n) = 1$, $r(1,n) = n^2$ and $r(n,n) = n!$. In general $r(k,n) = \binom{n}{k}^2 k!$.

Rook polynomials are related to Laguerre polynomials by the equation $\mathcal{R}_n(x) = n!x^n\mathcal{L}_n(-x^{-1})$. Therefore, as a direct consequence of the above theorem, we can also characterize $|\mathcal{E}(\mathbf{M}_n)|$ 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* $|\mathcal{E}(\mathbf{M}_n)| = \sum_{k=0}^{n+1} r'(k,n)$.

We conclude this section with another pleasant correspondence between the endomorphisms in $\mathcal{E}(\mathbf{M}_n)$ and $\mathcal{R}_n(x)$. Let $f : L \to L$ 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 $x = 1$.

Corollary 2. $\mathcal{R}_n(1) = |\{\ f \in \mathcal{E}(\mathbf{M}_n) \mid f \textit{ is non-reducing in } \mathbf{M}_n\ \}|$.

Table 1 illustrates the join-endomorphisms over the lattice $\mathbf{M}_n$ as a union $\bigcup_{i=1}^{4} \mathcal{F}_i$. Corollary 2 follows from the observation that the set of non-reducing functions in $\mathbf{M}_n$ is equal to $\mathcal{F}_4$ whose size is $\mathcal{R}_n(1)$ as shown in the following proof of Th. 1.

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

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

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

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

Suppose $f(\bot) = \bot$. Let J, K, H be disjoint possibly empty sets such that $I = J \cup K \cup H$ 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 $\bot$, respectively. More precisely, $\mathrm{Img}(f\restriction_J) = \{\top\}$, $\mathrm{Img}(f\restriction_K) \subseteq I$ and $\mathrm{Img}(f\restriction_H) = \{\bot\}$. Furthermore, for every f either (1) $f(\top) = \bot$, (2) $f(\top) \in I$ or (3) $f(\top) = \top$. We show that $f \notin \mathcal{E}(\mathbf{M}_n)$ for case (3), proofs of cases (1) and (2) are included in [13].

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 $e_1, e_2 \in H$ such that $f(e_1) = f(e_2) = \bot$. But then $f(e_1 \sqcup e_2) = f(\top) = \top \neq \bot = f(e_1) \sqcup f(e_2)$, hence $f \notin \mathcal{E}(\mathbf{M}_n)$.

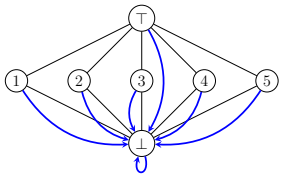
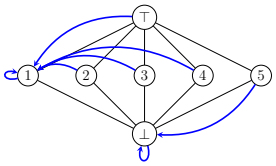
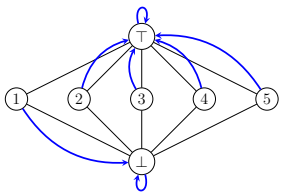
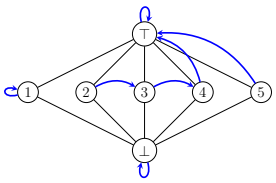
	
Let $\mathcal{F}_1$ be the family of functions f that for all $e \in \mathbf{M}_n$, $f(e) = \bot$.	Let $\mathcal{F}_2$ be the family of bottom preserving functions f such that for some $e, e' \in I$: (a) $f(\top) = e$, (b) $f(e') = \bot$ or $f(e') = e$, and (c) $f(e'') = e$ for all $e'' \in I \setminus \{e'\}$.
	
Let $\mathcal{F}_3$ be the family of top and bottom preserving functions f such that for some $e \in I$: (a) $f(e) = \bot$, and (b) $f(e') = \top$ for all $e' \in I \setminus \{e\}$.	Let $\mathcal{F}_4$ be the family of top and bottom preserving functions f that for some $J \subseteq I$: (a) $f(e) = \top$ for every $e \in J$, (b) $f\restriction_{I\setminus J}$ is injective, and (c) $\mathrm{Img}(f\restriction_{I\setminus J}) \subseteq I$.

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

Suppose $k > 0$. Assume $h = 0$. Notice that $K = I \setminus J$ and $\mathrm{Img}(f\restriction_K) \subseteq I$. Since f is a $\bot$ and $\top$ preserving function and it satisfies conditions (a) and (c) of $\mathcal{F}_4$ but $f \notin \mathcal{F}_4$, then f must violate condition (b). Thus $f\restriction_K$ is not injective. Then there are $a, b \in K$ such that $a \neq b$ but $f(a) = f(b)$. Then $f(a) \sqcup f(b) \neq \top = f(a \sqcup b)$. Consequently, $f \notin \mathcal{E}(\mathbf{M}_n)$.

Assume $h > 0$. There must be $e_1, e_2, e_3 \in I$ such that $f(e_1) = \bot$ 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, $f \notin \mathcal{E}(\mathbf{M}_n)$.

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 [13] for details. Here we prove that $|\mathcal{F}_4| = n!\mathcal{L}_n(-1)$.

Let $f \in \mathcal{F}_4$ and let $J \subseteq I$ be a possibly empty set such that $\mathrm{Img}(f\restriction_J) = \{\top\}$ and $\mathrm{Img}(f\restriction_{I\setminus J}) \subseteq I$, where $f\restriction_{I\setminus J}$ is an injective function. We shall call $j = |J|$.

For each of the $\binom{n}{j}$ possibilities for J, the elements of $I \setminus J$ are to be mapped to I by the injective function $f\restriction_{I\setminus J}$. The number of functions $f\restriction_{I\setminus J}$ is $\frac{n!}{j!}$. Therefore, $|\mathcal{F}_4| = \sum_{j=0}^{n} \binom{n}{j} \frac{n!}{j!}$. 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. □

4 Algorithms

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

Finding $\bigsqcap_{\mathcal{E}(L)} S$ may not be immediate. E.g., see $\bigsqcap_{\mathcal{E}(L)} S$ in Fig.1a for a small lattice of four elements and two join-endomorphisms. As already mentioned, a *naive approach* is to compute $\bigsqcap_{\mathcal{E}(L)} S$ by taking $\sigma_S(c) \stackrel{\text{def}}{=} \bigsqcap_L \{f(c) \mid f \in S\}$ 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 compute $\bigsqcap_{\mathcal{E}(L)} S$ can be obtained by generating the set $S' = \{g \mid g \in \mathcal{E}(L) \text{ and } g \sqsubseteq f \text{ for all } f \in S\}$ and taking its join. This approach works since $\bigsqcup S' = \bigsqcap_{\mathcal{E}(L)} S$ but as shown in Section 3, the size of $\mathcal{E}(L)$ can be super-polynomial for distributive lattices and exponential in general.

Nevertheless, one can use lattice properties to compute $\bigsqcap_{\mathcal{E}(L)} S$ efficiently. For distributive lattices, we use the inherent compositional nature of $\bigsqcap_{\mathcal{E}(L)} S$. For arbitrary lattices, we present an algorithm that uses the function σ_S in the naive approach to compute $\bigsqcap_{\mathcal{E}(L)} S$ 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.

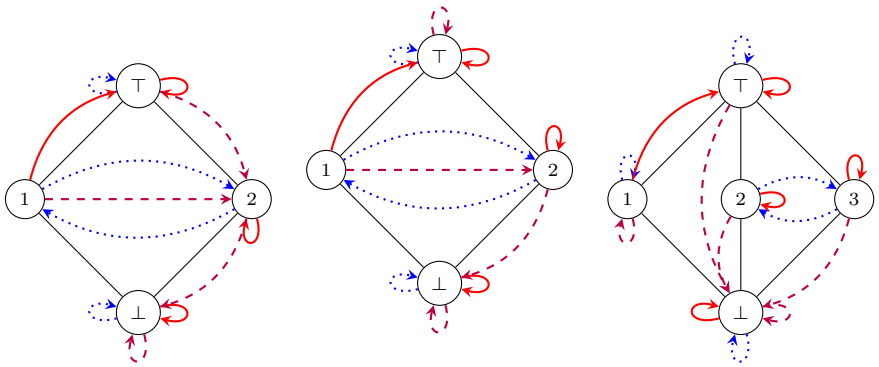
4.1 Meet of Join-Endomorphisms in Distributive Lattices

Here we shall illustrate some pleasant compositionality properties of the infima of join-endomorphisms that can be used for computing the join-endomorphism $\bigsqcap_{\mathcal{E}(L)} S$ in a finite distributive 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 $j \in J$.

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

The above lemma basically says that $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c)$ 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 $\delta_S \sqsupseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$ uses the fact that join-endomorphisms preserve joins. The proof that $\delta_S \sqsubseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$ proceeds by showing that δ_S is a lower bound in $\mathcal{E}(L)$ of S. Distributivity of the lattice L is crucial for this direction. In fact without it $\bigsqcap_{\mathcal{E}(L)} S = \delta_S$ does not necessarily hold as shown by the following counter-example.



(a) f_1: ⋯→, f_2: →, f: ⇢ (b) f_1: ⋯→, f_2: →, σ_S: ⇢ (c) f_1: ⋯→, f_2: →, δ_S: ⇢

Fig. 1: $S = \{f_1, f_2\} \subseteq \mathcal{E}(L)$. (a) $f = \bigsqcap_{\mathcal{E}(L)} S$. (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 $\mathbf{M}_3$: $\delta_S(1) \sqcup \delta_S(2) = 1 \neq \bot = \delta_S(1 \sqcup 2)$.

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

Naive Algorithm A_1. One could use Lemma 1 directly in the obvious way to provide an algorithm for $\bigsqcap_{\mathcal{E}(L)} S$ 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 $\bigsqcap_{\mathcal{E}(L)} S$ 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 $\bigsqcap_{\mathcal{E}(L)} S$ that can be used in a divide-and-conquer algorithm with lower time complexity.

Proposition 5. *Let L be a finite distributive lattice and $S = S_1 \cup S_2 \subseteq \mathcal{E}(L)$. Then $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c) = \bigsqcap_L \{\big(\bigsqcap_{\mathcal{E}(L)} S_1\big)(a) \sqcup \big(\bigsqcap_{\mathcal{E}(L)} S_2\big)(b) \mid a, b \in L \text{ and } a \sqcup b \sqsupseteq c\}$.*

The above proposition bears witness to the compositional nature of $\bigsqcap_{\mathcal{E}(L)} S$. It can be proven by replacing $\big(\bigsqcap_{\mathcal{E}(L)} S_1\big)(a)$ and $\big(\bigsqcap_{\mathcal{E}(L)} S_2\big)(b)$ by $\delta_{S_1}(a)$ and $\delta_{S_2}(b)$ using Lemma 1 (see [13]).

Naive Algorithm A_2. We can use Prop.5 to compute $\bigsqcap_{\mathcal{E}(L)} S$ with the following recursive procedure: 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 $\big(\bigsqcap_{\mathcal{E}(L)} S_1\big)(a) \sqcup \big(\bigsqcap_{\mathcal{E}(L)} S_2\big)(b)$ for every a, b such that $a \sqcup b \sqsupseteq c$. Then given $c \in L$, the time complexity of a *naive* implementation of the above procedure 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, $\bigsqcap_{\mathcal{E}(L)} S$ can be computed in $O(mn^{1+2\log_2 m})$.

The time complexity of the naive algorithm A_2 is better than that of A_1. However, by using a simple memoization technique to avoid repeating recursive calls and the following observations one can compute $\bigsqcap_{\mathcal{E}(L)} S$ in a much lower time complexity order.

4.2 Using Subtraction and Downsets to characterize $\bigsqcap_{\mathcal{E}(L)} S$

In what follows we show that $\bigsqcap_{\mathcal{E}(L)} S$ can be computed in $O(mn^2)$ for distributive lattices and, in particular, in $O(n + m \log n)$ for powerset lattices. To achieve this we use the subtraction operator from co-Heyting algebras and the notion of down set[4].

Subtraction Operator. Notice that in Prop.5 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 distributive lattices, and more generally in co-Heyting algebras [5], the *subtraction* operator $c \backslash a$ gives us exactly such a least element. The subtraction operator is uniquely determined by the property (*Galois connection*) $b \sqsupseteq c \backslash a$ iff $a \sqcup b \sqsupseteq c$ for all $a, b, c \in L$.

Down-sets. Besides using just $c \backslash a$ 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 $\big(\bigsqcap_{\mathcal{E}(L)} S_1\big)(a) \sqcup \big(\bigsqcap_{\mathcal{E}(L)} S_2\big)(c \backslash a) \sqsubseteq \big(\bigsqcap_{\mathcal{E}(L)} S_1\big)(a') \sqcup \big(\bigsqcap_{\mathcal{E}(L)} S_2\big)(c \backslash a')$.

The above observations lead us to the following theorem.

Theorem 2. *Let L be a finite distributive lattice and $S = S_1 \cup S_2 \subseteq \mathcal{E}(L)$. Then* $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c) = \bigsqcap_L \{\big(\bigsqcap_{\mathcal{E}(L)} S_1\big)(a) \sqcup \big(\bigsqcap_{\mathcal{E}(L)} S_2\big)(c \backslash a) \mid a \in \downarrow c\}$.

The above result can be used to derive a simple recursive algorithm that, given a finite distributive lattice L and $S \subseteq \mathcal{E}(L)$, computes $\bigsqcap_{\mathcal{E}(L)} S$ in worst-case time complexity $O(mn^2)$ where $m = |S|$ and $n = |L|$. We show this algorithm next.

4.3 Algorithms for Distributive Lattices

We first describe the algorithm DMeetApp that computes the value $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c)$. We then describe the algorithm DMeet that computes the function $\bigsqcap_{\mathcal{E}(L)} S$ by calling DMeetApp in a particular order to avoid repeating computations. We use the following definition to specify the calling order.

[4] Recall that we give time complexities in terms of the number of basic binary lattice operations (i.e., meets, joins and subtractions) performed during execution.

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. We use $\langle S, \Delta_1, \Delta_2\rangle$ 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 6. *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 $S \subseteq \mathcal{E}(L)$ where L is a distributive lattice. The recursive program DMeetApp(Δ, c) defined in Algorithm 1 computes $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c)$. 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.2. 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 $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c)$ where Δ is a bpt of $S \subseteq \mathcal{E}(L)$ and L is a finite distributive lattice. The global variable T is used as a lookup table.

```
1: procedure DMeetApp(Δ, c)                                   ▷ Δ = ⟨S, Δ1, Δ2⟩
2:     if IsNull(T[S, c]) then
3:         if S = {f} then
4:             T[S, c] ← f(c)
5:         else
6:             T[S, c] ← ⊓_L{DMeetApp(Δ1, a) ⊔ DMeetApp(Δ2, c\a) | a ∈ ↓c}.
```

Computing $\bigsqcap_{\mathcal{E}(L)} S$ for Distributive Lattices. Let us consider an execution of DMeetApp(Δ, c). From the definition of subtraction it follows that $c \backslash a \in {\downarrow}c$. Then for each recursive call DMeetApp(Δ', a') performed by an execution of DMeetApp(Δ, c) we have $a' \in {\downarrow}c$. This and the fact that T is initialized with `null` values not in L lead us the following simple observation.

Observation 3 *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 (meets, joins, substractions) performed by* DMeetApp(Δ, c) *is in $O(|\,{\downarrow}c\,|)$.*

Algorithm 2 DMeet(L, S, P). Given a finite distributive lattice L, $P \subseteq L$ and $S \subseteq \mathcal{E}(L)$, the algorithm computes $T[S, c] = \bigsqcap_{\mathcal{E}(L)} S(c)$ for each $c \in P$. Δ is a bpt of S and T is a global lookup table.

```
1: T[S', a] ← null                          ▷ for each a ∈ P and each node S' of Δ
2: for each S' in a post-order traversal sequence of Δ do   ▷ visit each S' of Δ in post-order
3:     for each c ∈ P in increasing order do   ▷ visit each c ∈ P in increasing order w.r.t L
4:         DMeetApp(Δ(S'), c)
```

DMeet(L, S, P). The values of $\big(\bigsqcap_{\mathcal{E}(L)} S\big)(c)$ for each $c \in P \subseteq L$ are computed by the program in Algo.2 as follows. To satisfy the assumption in Obs.3, it visits each node S' of Δ in *post-order* (i.e., before visiting a node it first visits its children). For each subtree $\Delta(S')$ of Δ, it calls DMeetApp$(\Delta(S'), c)$ for every $c \in P$ in *increasing order* with respect to the order of L: I.e., before calling DMeetApp$(\Delta(S'), c)$ it calls first DMeetApp$(\Delta(S'), c')$ for each $c' \in (P \cap {\downarrow}c) \setminus \{c\}$. The correctness of the call DMeet(L, S, P) follows from that of DMeetApp(Δ, c).

Complexity for Distributive Lattices. Assume that L is a distributive lattice of size n and that S is a subset of $\mathcal{E}(L)$ of size m. The above-mentioned traversals of Δ and P ensure that the assumption in Obs.3 is satisfied by each call of the form DMeetApp$(\Delta(S'), c)$ performed during the execution of DMeet(L, S, L). From Prop.6 we know that the number of iterations of the outer **for** is $2m - 1$. Clearly $|{\downarrow}c|$ and $|P|$ are both in $O(n)$. Thus, given S' we conclude from Obs.3 that the total number of operations from all calls of the form DMeetApp$(\Delta(S'), c)$, executed in the inner **for**, is in $O(n^2)$. The worst-case time complexity of DMeet(L, S, L) is then in $O(mn^2)$.

Complexity for Powerset Lattices. Assume that L is a powerset lattice. We can compute $\bigsqcap_{\mathcal{E}(L)} S$ in $O(n + m\log n)$ as follows. First call DMeet(L, S, P) where $P = J(L) \cup \{\bot\}$ and $J(L)$ is the set of *join-irreducible* elements (i.e., the singleton sets in this case) of L. Since $|J(L)| = \log_2 n$ and $|{\downarrow}c| = 2$ for every $c \in J(L)$, DMeet(L, S, P) can be performed in $O(m \log n)$. This produces $T[S, c] = \big(\bigsqcap_{\mathcal{E}(L)} S\big)(c)$ for each $c \in P$. To compute $T[S, e] = \big(\bigsqcap_{\mathcal{E}(L)} S\big)(e)$ for each $e \in L \setminus P$ in a total time of $O(n)$, visit each such an e in increasing order and set $T[S, e] = T[S, a] \sqcup T[S, b]$ for some $a, b \in {\downarrow}e \setminus \{e\}$ such that $e = a \sqcup b$. Since $e \notin P$ there must be a, b satisfying the above condition.

4.4 Algorithms for Arbitrary Lattices

The previous algorithm may fail to produce the $\bigsqcap_{\mathcal{E}(L)} S$ for non-distributive finite lattices. Nonetheless, for any arbitrary finite lattice L, $\bigsqcap_{\mathcal{E}(L)} S$ can be computed by successive approximations, starting with some self-map known to be smaller than each $f \in S$ and greater than $\bigsqcap_{\mathcal{E}(L)} S$. Assume a self-map $\sigma : L \to L$

such that $\sigma \sqsupseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$ and, for all $f \in S$, $\sigma \sqsubseteq_{\mathcal{E}} f$. A good starting point is $\sigma(u) = \bigsqcap\{f(u) \mid f \in S\}$, for all $u \in L$. By definition of $\sqcap$, σ is the biggest function under all functions in S, hence $\sigma \sqsupseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$. The program GMeet in Algorithm 3 computes decreasing upper bounds of $\bigsqcap_{\mathcal{E}(L)} S$ 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 $\sigma \sqsupseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$, as stated in Thm.4.

Theorem 4. *Let L be a finite lattice, $u, v \in L$, $\sigma : L \to L$ and $S \subseteq \mathcal{E}(L)$. Assume $\sigma \sqsupseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$ 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) $\sigma' \sqsupseteq_{\mathcal{E}} \bigsqcap_{\mathcal{E}(L)} S$.

Algorithm 3 GMeet finds $\sigma = \bigsqcap_{\mathcal{E}(L)} S$

```
1: σ(u) ← ⊓{f(u) | f ∈ S}                          ▷ for all u ∈ L
2: while u, v ∈ L ∧ σ(u) ⊔ σ(v) ≠ σ(u ⊔ v) do
3:     if σ(u) ⊔ σ(v) ⊏ σ(u ⊔ v) then               ▷ case (1)
4:         σ(u ⊔ v) ← σ(u) ⊔ σ(v)
5:     else                                         ▷ case (2)
6:         σ(u) ← σ(u) ⊓ σ(u ⊔ v)
7:         σ(v) ← σ(v) ⊓ σ(u ⊔ 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.4. 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 $\sigma = \bigsqcap_{\mathcal{E}(L)} S$.

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 $\texttt{Sup}_w = \{(u, v) \mid w = u \sqcup v \wedge \sigma(u) \sqcup \sigma(v) =$

Algorithm 4 GMEET+ finds $\sigma = \bigsqcap_{\mathcal{E}(L)} S$

```
1: σ(u) ← ⊓{f(u) | f ∈ S}                                   ▷ for all u ∈ L
2: Initialize Sup_w, Con_w, Fail_w, for all w
3: while w ∈ L such that (u, v) ∈ Con_w do                  ▷ some conflict set not empty
4:     Con_w ← Con_w \ {(u, v)}
5:     σ(w) ← σ(u) ⊔ σ(v)
6:     Fail_w ← Fail_w ∪ Sup_w                  ▷ all pairs previously in Sup_w are now failures
7:     Sup_w ← {(u, v)}
8:     checkSupports(w)                          ▷ for u ∈ L, verify property Sup_{w⊔u}
9:     while z ∈ L such that (x, y) ∈ Fail_z do             ▷ some failures set not empty
10:        Fail_z ← Fail_z \ {(x, y)}
11:        if σ(x) ≠ σ(x) ⊓ σ(z) then
12:            σ(x) ← σ(x) ⊓ σ(z)                                        ▷ σ(x) decreases
13:            Fail_x ← Fail_x ∪ Sup_x              ▷ all pairs in Sup_x are now failures
14:            Sup_x ← ∅
15:            checkSupports(x)                     ▷ for u ∈ L, verify property Sup_{x⊔u}
16:        if σ(y) ≠ σ(y) ⊓ σ(z) then
17:            σ(y) ← σ(y) ⊓ σ(z)                                        ▷ σ(y) decreases
18:            Fail_y ← Fail_y ∪ Sup_y              ▷ all pairs in Sup_y are now failures
19:            Sup_y ← ∅
20:            checkSupports(y)                     ▷ for u ∈ L, verify property Sup_{y⊔u}
21:        if σ(x) ⊔ σ(y) = σ(z) then
22:            Sup_z ← Sup_z ∪ {(x, y)}                                  ▷ (x, y) is now correct
23:        else
24:            Con_z ← Con_z ∪ {(x, y)}                                  ▷ (x, y) is now a conflict
```

$\sigma(w)\}$. We also have a conflicts set $\texttt{Con}_w = \{(u,v) \mid w = u \sqcup v \wedge \sigma(u) \sqcup \sigma(v) \sqsubset \sigma(w)\}$ and failures set $\texttt{Fail}_w = \{(u,v) \mid w = u \sqcup v \wedge \sigma(u) \sqcup \sigma(v) \not\sqsubseteq \sigma(w)\}$. Algorithm 4 updates σ as mentioned in Thm.4 and so maintains the invariant $\sigma \sqsupseteq \bigsqcap_{\mathcal{E}(L)} S$. An additional invariant is that, for all w, sets $\texttt{Sup}_w, \texttt{Con}_w, \texttt{Fail}_w$ are pairwise disjoint. When the outer loop finishes sets $\texttt{Con}_w$ and $\texttt{Fail}_w$ are empty (for all w) and thus every (u,v) belongs to $\texttt{Sup}_{u \sqcup v}$, i.e. the resulting $\sigma = \bigsqcap_{\mathcal{E}(L)} S$.

Auxiliary procedure checkSupports(u) identifies all pairs of the form $(u,x) \in \texttt{Sup}_{u \sqcup x}$ 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* $\texttt{Con}_w$ set or in the *failures* $\texttt{Fail}_w$ set. When a w is selected (for some (u,v) such that $u \sqcup v = w$) the following 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

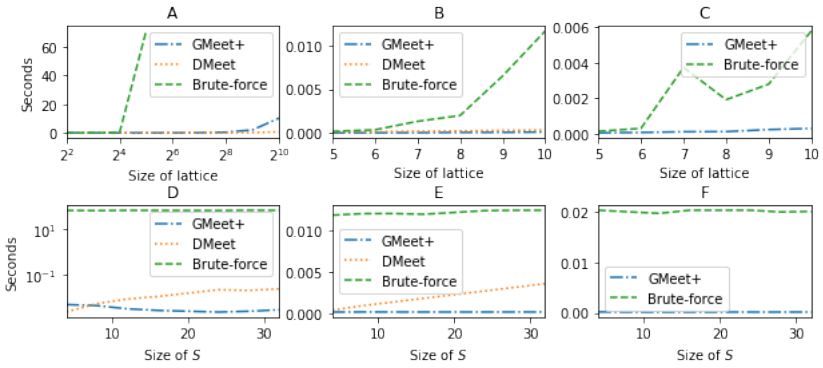


Fig. 2: Average performance time of GMeet+, DMeet and Brute-force. Plots A and D use $\mathbf{2}^n$ lattices, B and E distributive lattices, and C and F arbitrary (possibly non-distributive) lattices. Plots A-C have a fixed number of join-endomorphisms and plots D-F have a fixed lattice size.

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 $O(n^2 d)$. 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 dilation operators from Mathematical Morphology [2]. 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, the horizontal axis is the size of the lattice. In plots 2.D-F, the 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 $S \subseteq \mathcal{E}(L)$ 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. In all cases the lattices were randomly generated, and the parameters selected to showcase the difference between each algorithm with a sensible overall execution time. For a given lattice L and $S \subseteq \mathcal{E}(L)$, the brute-force algorithm explores the whole space $\mathcal{E}(L)$ to find all the join-endomorphism below each element of S and then computes the greatest of them.

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 $|S| = 4$

In particular, the measured spike in plot 2.C corresponds to the random lattice of seven elements with the size of $\mathcal{E}(L)$ 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 $\mathbf{2}^n$ (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 5.

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 [2,14,17], among others, complete lattices together with certain kinds of morphisms, such as *dilations*, defined as *join-endomorphisms* [14]. Our results give bounds about the number of all dilations over certain specific finite lattices and also efficient algorithms to compute their infima.

A typical application of MM is image processing. Consider the space $G = \mathbb{Z}^2$. A dilation [2] by $s_i \subseteq \mathcal{P}(G)$ is a function $\delta_{s_i} : \mathcal{P}(G) \to \mathcal{P}(G)$ such that $\delta_{s_i}(X) = \{x + e \mid x \in X \text{ and } e \in s_i\}$. The dilation $\delta_{s_i}(X)$ describes the interaction of an image X with the *structuring element* s_i. Intuitively, the dilation of X by s_i is the result of superimpose s_i on every activated pixel of X, with the center of s_i aligned with the corresponding pixel of X. Then, each pixel of every superimposed s_i is included in $\delta_{s_i}(X)$.

Let L be the powerset lattice for some finite set $D \subseteq G$. It turns out that the dilation $\bigsqcap_{\mathcal{E}(L)} S$ corresponds to the intersection of the structuring elements of the corresponding dilations in S. Fig.3 illustrates $\bigsqcap_{\mathcal{E}(L)} S$ for the two given dilations $\delta_{s_1}(I)$ and $\delta_{s_2}(I)$ with structuring elements s_1 and s_2 over the given image I.

5 Conclusions and Related Work

We have shown that given a lattice L of size n and a set $S \subseteq \mathcal{E}(L)$ of size m, $\bigsqcap_{\mathcal{E}(L)} S$ can be computed in the worst-case in $O(n + m\log n)$ binary lattice

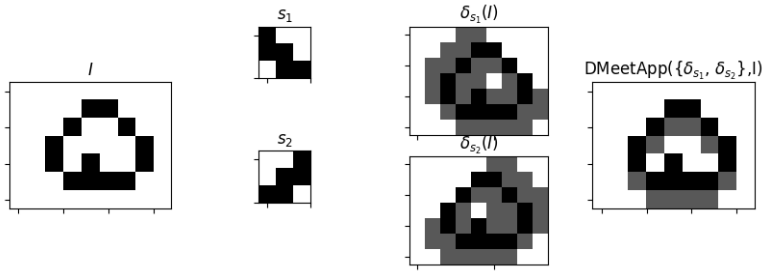


Fig. 3: Binary image I (on the left). Dilations δ_{s_1}, δ_{s_2} for structuring elements s_1, s_2. On the right $\left(\bigsqcap_{\mathcal{E}(L)}\{\delta_{s_1}, \delta_{s_2}\}\right)(I)$. New elements of the image after each operation in grey and black.

operations for powerset lattices, $O(mn^2)$ for lattices of sets, and $O(nm+n^3)$ for arbitrary lattices. We illustrated the experimental performance of our algorithms and a small example from mathematical morphology.

In [9] a bit-vector representation of a lattice is discussed. This work gives algorithms of logarithmic (in the size of the lattice) complexity for join and meet operations. These results count bit-vector operations. From [1] we know that $\mathcal{E}(L)$ is isomorphic to the downset of $(P \times P^{op})$, where P is the set of join-prime elements of L, and that this, in turn, is isomorphic to the set of order-preserving functions from $(P \times P^{op})$ to $\mathbf{2}$. Therefore, for the problem of computing $\bigsqcap_{\mathcal{E}(L)} S$, we get bounds $O(m \log_2(2^{(n^2)}) = O(mn^2)$ for set lattices and $O(m(\log_2 n)^2)$ for powerset lattices where $n = |L|$ and $m = |S|$. This, however, assumes a bit-vector representation of a lattice isomorphic to $\mathcal{E}(L)$. Computing this representation takes time and space proportional to the size of $\mathcal{E}(L)$ [9] which could be exponential as stated in the present paper. Notice that in our algorithms the input lattice is L instead of $\mathcal{E}(L)$.

We have stated the cardinality of the set of join-endomorphisms $\mathcal{E}(L)$ for significant families of lattices. To the best of our knowledge we are the first to establish the cardinality $(n+1)^2+n!\mathcal{L}_n(-1)$ for the lattice $\mathbf{M}_n$. The cardinalities $n^{\log_2 n}$ for power sets (boolean algebras) and $\binom{2n}{n}$ for linear orders can also be found in the lattice literature [1,10,16]. Our original proofs for these statements can be found in the technical report of this paper [13].

The lattice $\mathcal{E}(L)$ have been studied in [6]. The authors showed that a finite lattice L is distributive iff $\mathcal{E}(L)$ is distributive. A lower bound of $2^{2n/3}$ for the number of monotonic self-maps of any finite poset L is given in [4]. Nevertheless to the best of our knowledge, no other authors have studied the problem of determining the size $\mathcal{E}(L)$ nor algorithms for computing $\bigsqcap_{\mathcal{E}(L)} S$. 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 distributive 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 functions of a given lattice.

As future work we plan to explore in detail the applications of our work in mathematical morphology and computer music [15]. Furthermore, in the same spirit of [11] we have developed algorithms to generate distributive and arbitrary lattices. In our experiments, we observed that for every lattice L of size n we generated, $n^{\log_2 n} \leq |\mathcal{E}(L)| \leq (n+1)^2 + n!\mathcal{L}_n(-1)$ and if the generated lattice was distributive, $n^{\log_2 n} \leq |\mathcal{E}(L)| \leq \binom{2n}{n}$. We plan to establish if these inequalities hold for every finite lattice.

Acknowledgments. We are indebted to the anonymous referees and editors of RAMICS 2020 for helping us to improve one of the complexity bounds, some proofs, and the overall quality of the paper.

References

1. Birkhoff, G.: Lattice Theory. No. v. 25,pt. 2 in American Mathematical Society colloquium publications, American Mathematical Society (1967)
2. Bloch, I., Heijmans, H., Ronse, C.: Mathematical morphology. In: Aiello, M., Pratt-Hartmann, I., Van Benthem, J. (eds.) Handbook of Spatial Logics. pp. 857–944. Springer Netherlands (2007)
3. Davey, B.A., Priestley, H.A.: Introduction to lattices and order. Cambridge university press, 2nd edn. (2002)
4. Duffus, D., Rodl, V., Sands, B., Woodrow, R.: Enumeration of order preserving maps. Order **9**(1), 15–29 (1992)
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 **9**, 722–722 (1958)
7. Guzmán, M., Haar, S., Perchy, S., Rueda, C., Valencia, F.D.: Belief, knowledge, lies and other utterances in an algebra for space and extrusion. J. Log. Algebr. Meth. Program. **86**(1), 107–133 (2017)
8. Guzmán, M., Knight, S., Quintero, S., Ramírez, S., Rueda, C., Valencia, F.D.: Reasoning about Distributed Knowledge of Groups with Infinitely Many Agents. In: CONCUR 2019 - 30th International Conference on Concurrency Theory. vol. 29, pp. 1–29 (2019)
9. Habib, M., Nourine, L.: Tree structure for distributive lattices and its applications. Theoretical Computer Science **165**(2), 391–405 (1996)
10. Jipsen, P.: Relation algebras, idempotent semirings and generalized bunched implication algebras. In: Relational and Algebraic Methods in Computer Science. pp. 144–158. Springer International Publishing (2017)
11. Jipsen, P., Lawless, N.: Generating all finite modular lattices of a given size. Algebra universalis **74**(3), 253–264 (2015)

12. Knight, S., Palamidessi, C., Panangaden, P., Valencia, F.D.: Spatial and Epistemic Modalities in Constraint-Based Process Calculi. In: 23rd International Conference on Concurrency Theory. Lecture Notes in Computer Science, vol. 7454, pp. 317–332. Springer (2012)
13. Quintero, S., Ramírez, S., Rueda, C., Valencia, F.D.: Counting and Computing Join-Endomorphisms in Lattices. Research report, LIX, Ecole polytechnique ; INRIA Saclay - Ile-de-France (2019), https://hal.archives-ouvertes.fr/hal-02422624
14. Ronse, C.: Why mathematical morphology needs complete lattices. Signal Processing **21**(2), 129 – 154 (1990)
15. Rueda, C., Valencia, F.: On validity in modelization of musical problems by ccp. Soft Computing **8**(9), 641–648 (2004)
16. Santocanale, L.: On Discrete Idempotent Paths. In: Combinatorics on Words. vol. 11682, pp. 312–325. Springer (2019)
17. Stell, J.: Why mathematical morphology needs quantales. In: Wilkinson, M., Roerdink, J. (eds.) International Symposium on Mathematical Morphology, ISMM09. pp. 13–16. Institute for Mathematics and Computing Science, University of Groningen (2009)