Efficient Möbius Transformations and their applications to Dempster-Shafer Theory: Clarification and implementation ⋆

Maxime Chaveroche, Franck Davoine, Véronique Cherfaoui

▶ **To cite this version:**

Maxime Chaveroche, Franck Davoine, Véronique Cherfaoui. Efficient Möbius Transformations and their applications to Dempster-Shafer Theory: Clarification and implementation ⋆. 2021. hal-03429350

HAL Id: hal-03429350
https://hal.science/hal-03429350v1

Preprint submitted on 15 Nov 2021

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.

arXiv:2107.07359v1 [cs.CC] 15 Jul 2021

Efficient Möbius Transformations and their applications to Dempster-Shafer Theory: Clarification and implementation*

Maxime Chaveroche*, Franck Davoine, Véronique Cherfaoui

Sorbonne University Alliance, Université de technologie de Compiègne, CNRS, Heudiasyc, CS 60319 - 60203 Compiègne Cedex, France

Abstract

Dempster-Shafer Theory (DST) generalizes Bayesian probability theory, offering useful additional information, but suffers from a high computational burden. A lot of work has been done to reduce the complexity of computations used in information fusion with Dempster's rule. The main approaches exploit either the structure of Boolean lattices or the information contained in belief sources. Each has its merits depending on the situation. In this paper, we propose sequences of graphs for the computation of the zeta and Möbius transformations that optimally exploit both the structure of distributive semilattices and the information contained in belief sources. We call them the *Efficient Möbius Transformations* (EMT). We show that the complexity of the EMT is always inferior to the complexity of algorithms that consider the whole lattice, such as the *Fast Möbius Transform* (FMT) for all DST transformations. We then explain how to use them to fuse two belief sources. More generally, our EMTs apply to any function in any finite distributive lattice, focusing on a meet-closed or join-closed subset. This article extends our work published at the international conference on *Scalable Uncertainty Management* (SUM) [1]. It clarifies it, brings some minor corrections and provides implementation details such as data structures and algorithms applied to DST.

Keywords: Möbius Transform, Zeta Transform, Efficiency, distributive lattice, meet-closed subset, join-closed subset, Fast Möbius Transform, FMT, Dempster-Shafer Theory, DST, belief functions, efficiency, information-based, complexity reduction

1 Introduction

Dempster-Shafer Theory (DST) [2] is an elegant formalism that generalizes Bayesian probability theory. It is more expressive by giving the possibility for a source to represent its belief in the state of a variable not only by assigning credit directly to a possible state (strong evidence) but also by assigning credit to any subset (weaker evidence) of the set Ω of all possible states. This assignment of credit is called a *mass function* and provides meta-information to quantify the level of uncertainty about one's believes considering the way one established them, which is critical for decision making.

*This work was carried out and co-funded in the framework of the Labex MS2T and the Hauts-de-France region of France. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

*Corresponding author

Email address: name.surname@hds.utc.fr (Maxime Chaveroche*, Franck Davoine, Véronique Cherfaoui)

Nevertheless, this information comes with a cost: considering $2^{|\Omega|}$ potential values instead of only $|\Omega|$ can lead to computationally and spatially expensive algorithms. They can become difficult to use for more than a dozen possible states (e.g. 20 states in Ω generate more than a million subsets), although we may need to consider large frames of discernment (e.g. for classification or identification tasks). Moreover, these algorithms not being tractable anymore beyond a few dozen states means their performances greatly degrade before that, which further limits their application to real-time applications. To tackle this issue, a lot of work has been done to reduce the complexity of transformations used to combine belief sources with Dempster's rule [3]. We distinguish between two approaches that we call *powerset-based* and *evidence-based*.

The *powerset-based* approach concerns all algorithms based on the structure of the powerset 2^{Ω} of the frame of discernment Ω. They have a complexity dependent on $|\Omega|$. Early works [4, 5, 6, 7] proposed optimizations by restricting the structure of evidence to only singletons and their negation, which greatly restrains the expressiveness of DST. Later, a family of optimal algorithms working in the general case, i.e. the ones based on the *Fast Möbius Transform* (FMT) [8], was discovered. Their complexity is $O(|\Omega|.2^{|\Omega|})$ in time and $O(2^{|\Omega|})$ in space. It has become the de facto standard for the computation of every transformation in DST. Consequently, efforts were made to reduce the size of Ω to benefit from the optimal algorithms of the FMT. More specifically, [9] refers to the process of conditioning by the *combined core* (intersection of the unions of all *focal sets* of each belief source) and *lossless coarsening* (merging of elements of Ω which always appear together in focal sets). Also, Monte Carlo methods [9] have been proposed but depend on a number of trials that must be large and grows with $|\Omega|$, in addition to not being exact.

The *evidence-based* approach concerns all algorithms that aim to reduce the computations to the only subsets that contain information (*evidence*), called *focal sets*, which are usually far less numerous than $2^{|\Omega|}$. This approach, also refered to as the *obvious* one, implicitly originates from the seminal work of Shafer [2] and is often more efficient than the powerset-based one since it only depends on information contained in sources in a quadratic way. Doing so, it allows for the exploitation of the full potential of DST by enabling us to choose any frame of discernment, without concern about its size. Moreover, the evidence-based approach benefits directly from the use of approximation methods, some of which are very efficient [10]. Therefore, this approach seems superior to the FMT in most use cases, above all when $|\Omega|$ is large, where an algorithm with exponential complexity is just intractable.

It is also possible to easily find evidence-based algorithms computing all DST transformation, except for the conjunctive and disjunctive decompositions for which we recently proposed a method [11, 12].

However, since these algorithms rely only on the information contained in sources, they do not exploit the structure of the powerset to reduce the complexity, leading to situations in which the FMT can be more efficient if almost every subset contains information, i.e. if the number of focal sets tends towards $2^{|\Omega|}$ [9], all the most when no approximation method is employed.

In this paper, we fuse these two approaches into one, proposing new sequences of graphs, in the same fashion as the FMT, that are always more efficient than the FMT and can in addition benefit from evidence-based optimizations. We call them the *Efficient Möbius Transformations* (EMT). More generally, our approach applies to any function defined on a finite distributive lattice.

Outside the scope of DST, [13] is related to our approach in the sense that we both try to remove redundancy in the computation of the zeta and Möbius transforms on the subset lattice 2^{Ω}. However, they only consider the redundancy of computing the image of a subset that is known to be null beforehand. To do so, they only visit sets that are accessible from the focal

sets of lowest rank by successive unions with each element of Ω. Here, we demonstrate that it is possible to avoid far more computations by reducing them to specific sets so that each image is only computed once. These sets are the focal points described in [11, 12]. The study of their properties will be carried out in depth in an upcoming article [14]. Besides, our method is more general since it applies to any finite distributive lattice.

Furthermore, an important result of our work resides in the optimal computation of the zeta and Möbius transforms in any intersection-closed family F of sets from 2^Ω, i.e. with a complexity $O(|\Omega|.|F|)$. Indeed, in the work of [15] on the optimal computation of these transforms in any finite lattice L, they embedded L into the Boolean lattice 2^Ω, obtaining an intersection-closed family F as its equivalent, and found a meta-procedure building a circuit of size $O(|\Omega|.|F|)$ computing the zeta and Möbius transforms. However, they did not managed to build this circuit in less than $O(|\Omega|.2^{|\Omega|})$. Given F, our Theorem 3.2.2 in this paper directly computes this circuit with a complexity that can be as low as $O(|\Omega|.|F|)$ in some instances, while being much simpler.

This paper is organized as follows: Section 2 will present the elements on which our method is built. Section 3 will present our EMT. Section 4 will discuss their complexity and their usage both in general and in the case of DST. Finally, we will conclude this article with section 5.

2 Background of our method

Let $(P, \leq)$ be a finite[1] set partially ordered by $\leq$.

2.1 Zeta transform

The zeta transform $g : P \to \mathbb{R}$ of a function $f : P \to \mathbb{R}$ is defined as follows:

$$\forall y \in P, \quad g(y) = \sum_{x \leq y} f(x) \tag{1}$$

This can be extended to the multiplication as the *multiplicative zeta transform*:

$$\forall y \in P, \quad g(y) = \prod_{x \leq y} f(x)$$

Example 2.1.1. In DST, the implicability function b is defined as the zeta transform of the mass function m in $(2^\Omega, \subseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad b(B) = \sum_{A \subseteq B} m(A)$$

Example 2.1.2. In DST, the implicability function b is also the inverse of the multiplicative zeta transform of the disjunctive weight function v in $(2^\Omega, \subseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad b(B) = \prod_{A \subseteq B} v(A)^{-1}$$

[1]The following definitions hold for lower semifinite partially ordered sets as well, i.e. partially ordered sets such that the number of elements of P lower in the sense of $\leq$ than another element of P is finite. But for the sake of simplicity, we will only talk of finite partially ordered sets.

Example 2.1.3. In DST, the commonality function q is defined as the zeta transform of the mass function m in $(2^\Omega, \supseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad q(B) = \sum_{A \supseteq B} m(A)$$

Example 2.1.4. In DST, the commonality function q is also the inverse of the multiplicative zeta transform of the conjunctive weight function w in $(2^\Omega, \supseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad q(B) = \prod_{A \supseteq B} w(A)^{-1}$$

2.2 Möbius transform

The Möbius transform of g is f. It is defined as follows:

$$\forall y \in P, \quad f(y) = \sum_{x \leq y} g(x).\mu_{P,\leq}(x, y) \tag{2}$$

where $\mu_{P,\leq}$ is the Möbius function of $(P, \leq)$ (See [16]). There is also a multiplicative version with the same properties that can be seen as the exponential of the Möbius transform of $\log \circ g$:

$$\forall y \in P, \quad f(y) = \prod_{x \leq y} g(x)^{\mu_{P,\leq}(x,y)}$$

Example 2.2.1. In DST, the mass function m is the Möbius transform of the implicability function b in $(2^\Omega, \subseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad m(B) = \sum_{A \subseteq B} b(A).\mu_{2^\Omega,\subseteq}(A, B)$$

where for any $A, B \in 2^\Omega$, the Möbius function evaluates to $\mu_{2^\Omega,\subseteq}(A, B) = (-1)^{|B|-|A|}$, as recalled in [8].

Example 2.2.2. In DST, the disjunctive weight function v is the inverse of the multiplicative Möbius transform of the implicability function b in $(2^\Omega, \subseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad v(B) = \prod_{A \subseteq B} b(A)^{-\mu_{2^\Omega,\subseteq}(A,B)}$$

Example 2.2.3. In DST, the mass function m is the Möbius transform of the commonality function q in $(2^\Omega, \supseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad m(B) = \sum_{A \supseteq B} q(A).\mu_{2^\Omega,\supseteq}(A, B)$$

where for any $A, B \in 2^\Omega$, the Möbius function also evaluates to $\mu_{2^\Omega,\supseteq}(A, B) = (-1)^{|B|-|A|}$.

Example 2.2.4. In DST, the conjunctive weight function w is the inverse of the multiplicative Möbius transform of the commonality function q in $(2^\Omega, \subseteq)$, i.e.:

$$\forall B \in 2^\Omega, \quad w(B) = \prod_{A \supseteq B} q(A)^{-\mu_{2^\Omega,\supseteq}(A,B)}$$

2.3 Sequence of graphs and computation of the zeta transform

To yield $g(y)$ for some $y \in P$, we must sum all values $f(x)$ such that $x \leq y$. Our objective is to do it in the minimum number of operations, i.e. to minimize the number of terms to sum. If we only compute $g(y)$ alone, we have to pick every element in $\{x \in P \ / \ x \leq y\}$ and sum their associated values through f. However, if we compute $g(y)$ for all elements $y \in P$ at once, we can organize and mix these computations so that partial sums are reused for more than one value through g. Indeed, for any element $y \in P$, if there is an element $z \in P$ such that $y \leq z$, we have $g(z) = g(y) + \sum_{\substack{x \leq z \\ x \not\leq y}} f(x)$. So, we want to recursively build partial sums so that we can get the full sum on each $g(y)$ by only summing the values on some elements from $\{x \in P \ / \ x \leq y\}$. In other words, we would like to define an ordered sequence of transformations computing g from f.

Let us adopt the formalism of graph theory. Let $G_{\leq}$ be a directed acyclic graph in which the set of its nodes matches P and each arrow is directed by $\leq$. Let $E_{\leq}$ be the set of its arrows. We have $E_{\leq} = \{(x, y) \in P^2 \ / \ x \leq y\}$ and $G_{\leq} = (P, E_{\leq})$. Thus, computing $g(y)$ alone is equivalent to visiting the node y from all nodes x of $G_{\leq}$ such that there is an arrow $(x, y) \in E_{\leq}$. Each "visit" to node y from a node x corresponds to the computation of the operation $f(x) + \cdot$, where $\cdot$ represents the current state of the sum associated with y. Thus, $G_{\leq}$, combined with the binary operator $f(\cdot) + \cdot$, describes the transformation of 0 into g. More concisely, we will equivalently initialize our algorithm with values through f instead of 0 and exploit the combination of $G_{<}$ and $+$. We will say that the transformation $(G_{<}, f, +)$ *computes the zeta transform of* f *in* $(P, \leq)$. In the end, we want to minimize the number of "visits" to be made to all y, i.e. we want to minimize the total number of arrows to follow to compute every $g(y)$. Therefore, the question is: Is there an ordered sequence of graphs that can compute g from f with less arrows in total than $(G_{<}, f, +)$?

Let I_P be the set containing all identity arrows of $G_{\leq}$, i.e. $I_P = \{ \ (x, y) \in P^2 \ / \ x = y\}$. Consider that all elements $y \in P$ are initialized with $f(y)$. We are interested in finding a sequence of graphs that is equivalent to the arrows of $G_{<}$. Let $(H_i)_{i \in [\![1,n]\!]}$ be a sequence of n directed acyclic graphs $H_i = (P, E_i)$. We will note $((H_i)_{i \in [\![1,n]\!]}, f, +)$ the computation that transforms f into h_1 through the arrows of E_1, then transforms h_1 into h_2 through the arrows of E_2, and so on until the transformation of h_{n-1} into h_n through the arrows of E_n. We ignore all identity arrows in these computations. So, this sequence of graphs requires us to consider $|E_1| + |E_2| + \cdots + |E_n|$ arrows, but transforms f into h_n in $|E_1 \backslash I_P| + |E_2 \backslash I_P| + \cdots + |E_n \backslash I_P|$ operations.

Proposition 2.3.1. Let $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$. One particular sequence of interest is $(H_i)_{i \in [\![1,n]\!]}$, where $H_i = (2^{\Omega}, E_i)$ and:

$$E_i = \{(X, Y) \in 2^{\Omega} \times 2^{\Omega} / \ Y = X \cup \{\omega_i\}\}.$$

This sequence computes the same zeta transformations as $G_{\subset} = (2^{\Omega}, E_{\subset})$, where $E_{\subset} = \{(X, Y) \in 2^{\Omega} \times 2^{\Omega} \ / \ X \subset Y\}$.

Example 2.3.1. Let us say that $\Omega = \{a, b, c\}$. Crossing ignored arrows (i.e. identity arrows), we have:

- $E_1 = \{(X, Y) \in 2^{\Omega} \times 2^{\Omega} / \ Y = X \cup \{a\}\} = \{$

  $(\emptyset, \{a\}),$ ~~$(\{a\}, \{a\})$~~, $(\{b\}, \{a, b\}),$ ~~$(\{a, b\}, \{a, b\})$~~,
  $(\{c\}, \{a, c\}),$ ~~$(\{a, c\}, \{a, c\})$~~, $(\{b, c\}, \Omega),$ ~~(Ω, Ω)~~

  $\}$

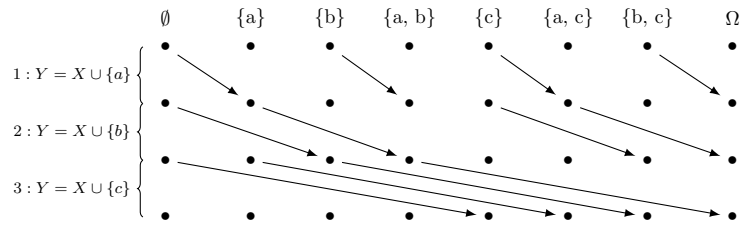

Figure 1: Illustration representing the paths generated by the arrows contained in the sequence $(H_i)_{i\in[\![1,3]\!]}$, where $H_i = (2^\Omega, E_i)$ and $E_i = \{(X,Y) \in 2^\Omega \times 2^\Omega / \ Y = X \cup \{\omega_i\}\}$ and $\Omega = \{a,b,c\}$. This sequence computes the same zeta transformations as $G_\subset = (2^\Omega, E_\subset)$, where $E_\subset = \{(X,Y) \in 2^\Omega \times 2^\Omega \ / \ X \subset Y\}$. A dot represents the node of its column. Its row i corresponds to both the tail of a potential arrow in H_i and the head of a potential arrow in H_{i-1}. The last row corresponds to the heads of all potential arrows that could be in H_3. The arrows represents the actual arrows in each graph H_i. Identity arrows are ignored in computations and not displayed here for the sake of clarity. If they were, there would be vertical arrows in every column, linking each dot in row i to the next dot of same node in row $i+1$. This representation is derived from the one used in [8].

- $E_2 = \{(X,Y) \in 2^\Omega \times 2^\Omega / \ Y = X \cup \{b\}\} = \{$

$$(\emptyset,\{b\}),\ (\{a\},\{a,b\}),\ \cancel{(\{b\},\{b\})},\ \cancel{(\{a,b\},\{a,b\})},$$
$$(\{c\},\{b,c\}),\ (\{a,c\}),\Omega),\ \cancel{(\{b,c\},\{b,c\})},\ \cancel{(\Omega,\Omega)}$$
$$\}$$

- $E_3 = \{(X,Y) \in 2^\Omega \times 2^\Omega / \ Y = X \cup \{c\}\} = \{$

$$(\emptyset,\{c\}),\ (\{a\},\{a,c\}),\ (\{b\},\{b,c\}),\ (\{a,b\},\Omega),$$
$$\cancel{(\{c\},\{c\})},\ \cancel{(\{a,c\}),\{a,c\})},\ \cancel{(\{b,c\},\{b,c\})},\ \cancel{(\Omega,\Omega)}$$
$$\}$$

Fig. 1 illustrates this sequence. Check that, after execution of $((H_i)_{i\in[\![1,n]\!]}, f, +)$, each element y of 2^Ω is associated with the sum $\sum_{x\subseteq y} f(x)$. For instance, let us take a look at Ω. Initially, each element of 2^Ω is associated with its value through f. Then, at step 1, we can see that the value on Ω is summed with $f(\{b,c\})$. At step 2, it is summed with $h_1(\{a,c\})$, which is equal to $f(\{c\})+f(\{a,c\})$, following step 1. Finally, at step 3, it is summed with $h_2(\{a,b\})$, which is equal to $h_1(\{a\})+h_1(\{a,b\})$ following step 2, which is itself equal to $f(\emptyset)+f(\{a\})+f(\{b\})+f(\{a,b\})$, following step 1. Gathering all these terms, we get that $h_3(\Omega) = f(\Omega) + f(\{b,c\}) + f(\{c\}) + f(\{a,c\}) + f(\emptyset) + f(\{a\}) + f(\{b\}) + f(\{a,b\})$.

Proposition 2.3.2. The dual of this particular sequence in $(2^\Omega, \supseteq)$ is $(H_i)_{i\in[\![1,n]\!]}$, where $H_i = (2^\Omega, E_i)$ and:

$$E_i = \{(X,Y) \in 2^\Omega \times 2^\Omega / \ X = Y \cup \{\omega_i\}\}.$$

This sequence computes the same zeta transformations as $G_\supset = (2^\Omega, E_\supset)$, where $E_\supset = \{(X,Y) \in 2^\Omega \times 2^\Omega \ / \ X \supset Y\}$.

Example 2.3.2. Fig. 2 illustrates the dual sequence for zeta transforms in $(2^\Omega, \supseteq)$ and $\Omega = \{a,b,c\}$.

Remark. These two sequences of graphs are the foundation of the *Fast Möbius Transform* (FMT) algorithms. Their execution is $O(n.2^n)$ in time and $O(2^n)$ in space. As we can see, the FMT presented here proposes two transformations $((H_i)_{i\in[\![1,n]\!]}, f, +)$ that computes the same

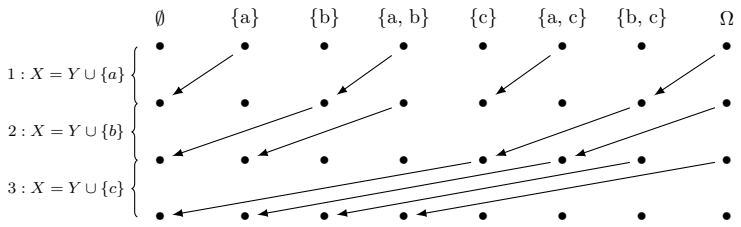

Figure 2: Illustration representing the paths generated by the arrows contained in the sequence $(H_i)_{i\in[\![1,3]\!]}$, where $H_i = (2^\Omega, E_i)$ and $E_i = \{(X,Y) \in 2^\Omega \times 2^\Omega / \ X = Y \cup \{\omega_i\}\}$ and $\Omega = \{a,b,c\}$. This sequence computes the same zeta transformations as $G_\supset = (2^\Omega, E_\supset)$, where $E_\supset = \{(X,Y) \in 2^\Omega \times 2^\Omega \ / \ X \supset Y\}$.

transformation as respectively $(G_\subset, f, +)$ and $(G_\supset, f, +)$ for any function $f : 2^\Omega \to \mathbb{R}$. The authors proved them to be the optimal transformations for any set Ω, i.e. the one that uses the fewest arrows, independently of the function f to be considered. This means that they do not take into account the neutral values of f for the operator $+$, i.e. where f evaluates to 0, contrary to our approach. This is why our method is able to feature a lower complexity than this *optimal* FMT.

More generally, Theorem 3 of [8] defines a necessary and sufficient condition to verify that a transformation $((H_i)_{i\in[\![1,n]\!]}, f, +)$ computes (ignoring identity arrows) the same transformation as $(G_<, f, +)$. It is stated in our terms as follows:

Theorem 2.3.1. *Let $(H_i)_{i\in[\![1,n]\!]}$ be a sequence of directed acyclic graphs $H_i = (P, E_i)$. Let us pose $A_i = E_i \cup I_P$. The transformation $((H_i)_{i\in[\![1,n]\!]}, f, +)$ computes (ignoring identity arrows) the same transformation as $(G_<, f, +)$ if and only if each set of arrows satisfies $E_i \subseteq E_\leq$ and every arrow $e \in E_\leq$ can be decomposed as a unique path $(e_1, e_2, \ldots, e_n) \in A_1 \times A_2 \times \cdots \times A_n$, where the tail of e_1 is the tail of e and the head of e_n is the head of e. Recall that a path is a sequence of arrows in which $\forall i \in [\![1, n-1]\!]$, the head of e_i is the tail of e_{i+1}.*

Example 2.3.3. Let us prove Proposition 2.3.1. We had $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$. The sequence of graphs $(H_i)_{i\in[\![1,n]\!]}$ computes the same zeta transformations as $G_\subset = (2^\Omega, E_\subset)$, where $E_\subset = \{(X,Y) \in 2^\Omega \times 2^\Omega \ / \ X \subset Y\}$ if:

$$E_i = \{(X,Y) \in 2^\Omega \times 2^\Omega / \ Y = X \cup \{\omega_i\}\},$$

where $i \in \{1, \ldots, n\}$ and $H_i = (2^\Omega, E_i)$.

Proof. Obviously, for any $i \in [\![1,n]\!]$, we have $E_i \subseteq E_\subseteq$. In addition, each set $X \subseteq \Omega$ is composed by definition of at most n elements from Ω. Thus, it is possible to reach in at most n steps from X any set $Y \subseteq \Omega$ such that $X \subseteq Y$ with identity arrows and arrows where the head is just the tail with exactly one other element from Ω, i.e. with identity arrows or arrows of $(H_i)_{i\in[\![1,n]\!]}$. Moreover, since each step corresponds to a distinct element from Ω, there is exactly one path from X to Y: at each step corresponding to the elements in $Y \backslash X$, follow the arrow that adds an element to the set reached in the previous step. Only identity arrows can be followed in the steps corresponding to the elements in $X \cap Y$. Otherwise, there would be an element missing or in excess relatively to Y at the end of step n, which means that Y could not be reached. Therefore, according to Theorem 2.3.1, the transformation $((H_i)_{i\in[\![1,n]\!]}, f, +)$ computes the same transformation as $(G_\subset, f, +)$ for any function $f : 2^\Omega \to \mathbb{R}$. ■

In addition, for a sequence of graphs computing zeta transforms in $(P, \leq)$, reversing its paths yields a sequence of graphs computing zeta transforms in $(P, \geq)$. Hence Proposition 2.3.2.

2.4 Sequence of graphs and computation of the Möbius transform

Now, consider that we want to find a sequence of graphs that undoes the previous computation. We want to transform g into f, i.e. the Möbius transform of g in $(P, \leq)$. For this, notice that for any step i in the transformation $((H_i)_{i\in[\![1,n]\!]}, f, +)$, we have for each node $y \in P$,

$$h_i(y) = h_{i-1}(y) + \sum_{(x,y)\in E_i\backslash I_P} h_{i-1}(x) \quad \Leftrightarrow \quad h_{i-1}(y) = h_i(y) - \sum_{(x,y)\in E_i\backslash I_P} h_{i-1}(x).$$

So, as long as we know all $h_{i-1}(x)$ for all arrows $(x, y) \in E_i\backslash I_P$ at each step i and for all $y \in P$, we can simply reverse the order of the sequence $(H_i)_{i\in[\![1,n]\!]}$ and use the operator - instead of +. If this is verified, then $((H_{n-i+1})_{i\in[\![1,n]\!]}, g, -)$ computes the Möbius transform of g in $(P, \leq)$. This condition can be translated as follows: for every arrow $(x, y) \in E_i\backslash I_P$, we have $h_i(x) = h_{i-1}(x)$. This condition is equivalent to stating that for every arrow $(x, y) \in E_i\backslash I_P$, there is no arrow (w, x) in $E_i\backslash I_P$.

Theorem 2.4.1. *Let $(H_i)_{i\in[\![1,n]\!]}$ be a sequence of directed acyclic graphs $H_i = (P, E_i)$. Let h_n be the function resulting from the transformation $((H_i)_{i\in[\![1,n]\!]}, f, +)$, ignoring identity arrows. If for every arrow $(x, y) \in E_i\backslash I_P$, there is no arrow (w, x) in $E_i\backslash I_P$, then $((H_{n-i+1})_{i\in[\![1,n]\!]}, h_n, -)$ yields the initial function f.*

Thus, if $((H_i)_{i\in[\![1,n]\!]}, f, +)$ computes the zeta transform g of f in $(P, \leq)$ and Theorem 2.4.1 is satisfied, then $((H_{n-i+1})_{i\in[\![1,n]\!]}, g, -)$ *computes the Möbius transform f of g in $(P, \leq)$.*

2.4.1 Application to the powerset lattice 2^Ω (FMT)

Consider again the sequence $(H_i)_{i\in[\![1,n]\!]}$, from the application of section 2.3.1, that computes the zeta transform of f in $(2^\Omega, \subseteq)$. If there is an arrow $(X, Y) \in E_i\backslash I_{2^\Omega}$, then $\omega_i \notin X$. This means that there is no set W in 2^Ω such that $W \cup \{\omega_i\} = X$, and so no arrow (W, X) in $E_i\backslash I_{2^\Omega}$. Thus, according to Theorem 2.4.1, $((H_{n-i+1})_{i\in[\![1,n]\!]}, h_n, -)$ computes the Möbius transformation of $((H_i)_{i\in[\![1,n]\!]}, f, +)$, i.e. the function f. Furthermore, given that Ω is a set and that each graph H_i concerns an element ω_i, independently from the others, any indexing (order) in the sequence $(H_i)_{i\in[\![1,n]\!]}$ computes the zeta transformation. This implies that any order in $((H_{n-i+1})_{i\in[\![1,n]\!]}, h_n, -)$ computes the Möbius transformation of $((H_i)_{i\in[\![1,n]\!]}, f, +)$. In particular, $((H_i)_{i\in[\![1,n]\!]}, h_n, -)$ computes the same transformation as $((H_{n-i+1})_{i\in[\![1,n]\!]}, h_n, -)$.

Example 2.4.1. Let us say that $\Omega = \{a, b, c\}$. We want to compute the Möbius transform f of g in $(2^\Omega, \subseteq)$. Each arrow set E_i has already been computed in Example 2.3.1. Fig. 1 illustrates any transformation based on $(H_i)_{i\in[\![1,n]\!]}$, including $((H_i)_{i\in[\![1,n]\!]}, h_n, -)$. Check that, after execution of $((H_i)_{i\in[\![1,n]\!]}, h_n, -)$, each element y of 2^Ω is associated with $f(y)$. For instance, let us take a look again at Ω. Initially, each element of 2^Ω is associated with its value through g. Then, at step 1, we can see that to the value on Ω is subtracted $g(\{b, c\})$. At step 2, to $h_1(\Omega)$ is subtracted $h_1(\{a, c\})$, which is equal to $g(\{a, c\}) - g(\{c\})$, following step 1. Finally, at step 3, to $h_2(\Omega)$ is subtracted $h_2(\{a, b\})$, which is equal to $h_1(\{a, b\}) - h_1(\{a\})$ following step 2, which is itself equal to $g(\{a, b\}) - g(\{b\}) - (g(\{a\}) - g(\emptyset))$, following step 1. Gathering all these terms, we get that

$$\begin{aligned} h_3(\Omega) &= g(\Omega) - g(\{b, c\}) - [\ g(\{a, c\}) - g(\{c\})\ ] - [\ g(\{a, b\}) - g(\{b\}) - [\ g(\{a\}) - g(\emptyset)\ ]\ ] \\ &= g(\Omega) - g(\{b, c\}) - g(\{a, c\}) + g(\{c\}) - g(\{a, b\}) + g(\{b\}) + g(\{a\}) - g(\emptyset) \\ &= \sum_{X\subseteq\Omega} g(x).(-1)^{|\Omega|-|X|}. \end{aligned}$$

As recalled in [8], the function that associates to each couple $(X, Y) \in 2^\Omega \times 2^\Omega$ the value $(-1)^{|Y|-|X|}$ is the Möbius function μ in $(2^\Omega, \subseteq)$. So, according to Eq. 2, we have $h_3(\Omega) = f(\Omega)$.

2.5 Order theory

Let $(P, \leq)$ be a set partially ordered by the relation $\leq$.

2.5.1 Meet / join

Let S be a subset of P. If it is unique, the greatest element of P that is less than all the elements of S is called the meet (or infimum) of S. It is noted $\bigwedge S$. If $S = \{x, y\}$, we may also note it with the binary operator $\wedge$ such that $x \wedge y$. If it is unique, the least element of P that is greater than all the elements of S is called the join (or supremum) of S. It is noted $\bigvee S$. If $S = \{x, y\}$, we may also note it with the binary operator $\vee$ such that $x \vee y$.

Example 2.5.1. In $(2^\Omega, \subseteq)$, the meet operator $\wedge$ is the intersection operator $\cap$, while the join operator $\vee$ is the union operator $\cup$.

2.5.2 *Lattice / semi-lattice*

If any non-empty subset of P has a join, we say that P is an upper semilattice. If any non-empty subset of P has a meet, we say that P is a lower semilattice. When P is both, we say that P is a lattice.

Example 2.5.2. In $(2^\Omega, \subseteq)$, any non-empty subset S has an intersection and a union in 2^Ω. They respectively represent the common elements of the sets in S and the cumulative elements of all the sets in S. Thus, 2^Ω is a lattice.

2.5.3 Irreducible elements

In any partially ordered set, there are bottom elements such that they cannot be described as the join of two lesser elements. Such irreducible elements are called the *join-irreducible elements of* P if they are not equal to the global minimum of P. We will note ${}^\vee\mathcal{I}(P)$ the set of all join-irreducible elements of P. Since none of them is $\bigwedge P$, this means that the join of any two join-irreducible elements yields a non-join-irreducible element of P. In fact, if P is an upper semilattice (or a lattice), it is known that the join of all possible non-empty subset of ${}^\vee\mathcal{I}(P)$ yields all elements of P, except $\bigwedge P$. Formally, we write that all join-irreducible element i verifies $i \neq \bigwedge P$ and for all elements $x, y \in P$, if $x < i$ and $y < i$, then $x \vee y < i$.

Dually, in any partially ordered set, there are top elements such that they cannot be described as the meet of two greater elements. Such irreducible elements are called the *meet-irreducible elements of* P if they are not equal to the global maximum of P. We will note ${}^\wedge\mathcal{I}(P)$ the set of all meet-irreducible elements of P. Since none of them is $\bigvee P$, this means that the meet of any two meet-irreducible elements yields a non-meet-irreducible element of P. In fact, if P is a lower semilattice (or a lattice), it is known that the meet of all possible non-empty subset of ${}^\wedge\mathcal{I}(P)$ yields all elements of P, except $\bigvee P$. Formally, we write that all meet-irreducible element i verifies $i \neq \bigvee P$ and for all elements $x, y \in P$, if $x > i$ and $y > i$, then $x \wedge y > i$.

Example 2.5.3. In $(2^\Omega, \subseteq)$, the join-irreducible elements are the singletons $\{\omega\}$, where $\omega \in \Omega$. The meet-irreducible elements are their complement $\overline{\{\omega\}} = \Omega \backslash \{\omega\}$, where $\omega \in \Omega$.

2.5.4 Distributive lattice

A distributive lattice L is a lattice that satisfies the distributive law:

$$\forall x, y, z \in L, \quad (x \wedge y) \vee (x \wedge z) = x \wedge (y \vee z) \tag{3}$$

Since $(x \wedge z) \vee z = z$, this condition is equivalent to:

$$\forall x, y, z \in L, \quad (z \vee y) \wedge (z \vee x) = z \vee (y \wedge x) \tag{4}$$

Example 2.5.4. In the powerset lattice 2^Ω, it holds for any sets $A, B, C \in 2^\Omega$ that $(A \cap B) \cup (A \cap C) = A \cap (B \cup C)$. Thus, the lattice 2^Ω is a distributive lattice.

2.5.5 Sublattice

A sublattice S is simply a subset of a lattice L that is itself a lattice, with the same meet and join operations as L. This means that for any two elements $x, y \in S$, we have both $x \vee y \in S$ and $x \wedge y \in S$, where $\vee$ and $\wedge$ are the join and meet operators of L.

2.5.6 Upset / down set

An upset (or upward closed set) S is a subset of P such that all elements in P greater than at least one element of S is in S. The upper closure of an element $x \in P$ is noted $\uparrow x$ or $x^{\uparrow P}$ (when the encompassing set has to be specified). It is equal to $\{y \in P \ / \ x \leq y \ \}$.

Dually, a down set (or downward closed set) S is a subset of P such that all elements in P less than at least one element of S is in S. The lower closure of an element $x \in P$ is noted $\downarrow x$ or $x^{\downarrow P}$ (when the encompassing set has to be specified). It is equal to $\{y \in P \ / \ x \geq y \ \}$.

2.6 Support elements and focal points

Let $f : P \to \mathbb{R}$.

2.6.1 Support of a function in P

The support $\text{supp}(f)$ of a function $f : P \to \mathbb{R}$ is defined as $\text{supp}(f) = \{x \in P \ / \ f(x) \neq 0\}$.

Example 2.6.1. In DST, the set containing the focal sets of a mass function m is $\text{supp}(m)$.

2.6.2 Focal points

We note ${}^\vee\text{supp}(f)$ the smallest join-closed subset of P containing $\text{supp}(f)$, i.e.:

$${}^\vee\text{supp}(f) = \left\{ \bigvee S \ / \ S \subseteq \text{supp}(f), \ S \neq \emptyset \right\}$$

We note ${}^\wedge\text{supp}(f)$ the smallest meet-closed subset of P containing $\text{supp}(f)$, i.e.:

$${}^\wedge\text{supp}(f) = \left\{ \bigwedge S \ / \ S \subseteq \text{supp}(f), \ S \neq \emptyset \right\}$$

The set containing the *focal points* $\mathring{\mathcal{F}}$ of a mass function m from [11, 12] for the conjunctive weight function is ${}^{\wedge}\text{supp}(m)$. For the disjunctive weight function, we use their dual focal points, defined by ${}^{\vee}\text{supp}(m)$.

It has been proven in [11, 12] that the image of 2^{Ω} through the conjunctive weight function can be computed without redundancies by only considering the focal points ${}^{\wedge}\text{supp}(m)$ in the definition of the multiplicative Möbius transform of the commonality function. The image of all set in $2^{\Omega}\backslash{}^{\wedge}\text{supp}(m)$ through the conjunctive weight function is 1. In the same way, the image of any set in $2^{\Omega}\backslash{}^{\wedge}\text{supp}(m)$ through the commonality function is only a duplicate of the image of a focal point in ${}^{\wedge}\text{supp}(m)$. Its image can be recovered by searching for its smallest focal point superset in ${}^{\wedge}\text{supp}(m)$. The same can be stated for the disjunctive weight function regarding the implicability function and ${}^{\vee}\text{supp}(m)$.

In fact, as generalized in [14], for any function $f : P \to \mathbb{R}$, its focal points ${}^{\wedge}\text{supp}(f)$ are sufficient to define its zeta and Möbius transforms in $(P, \geq)$, and ${}^{\vee}\text{supp}(f)$ is sufficient to define its zeta and Möbius transforms in $(P, \leq)$.

However, considering the case where P is a finite lattice, naive algorithms that only consider ${}^{o}\text{supp}(f)$, where $o \in \{\vee, \wedge\}$ have upper bound complexities in $O(|{}^{o}\text{supp}(f)|^2)$, which may be worse than the optimal complexity $O(|{}^{\vee}\mathcal{I}(P)|.|P|)$ of a procedure that considers the whole lattice P. In this paper, we propose computing schemes for $g(S)$, where ${}^{o}\text{supp}(f) \subseteq S \subseteq P$ and g is the zeta transform of f in $(P, \leq)$, with complexities always less than $O(|{}^{\vee}\mathcal{I}(P)|.|P|)$, provided that P is a finite distributive lattice. We also provide schemes for computing the Möbius transform $f(S)$ of g in $(P, \leq)$.

3 Our Efficient Möbius Transformations

In this section, we consider a function $f : P \to \mathbb{R}$ where P is a finite distributive lattice (e.g. the powerset lattice 2^{Ω}). We present here the sequences of graphs that can be exploited to compute our so-called *Efficient Möbius Transformations*. Theorem 3.2.1 describes a way of computing the zeta and Möbius transforms of a function based on the smallest sublattice ${}^{\mathcal{L}}\text{supp}(f)$ of P containing both ${}^{\wedge}\text{supp}(f)$ and ${}^{\vee}\text{supp}(f)$, which is defined in Proposition 3.1.3. Theorem 3.2.2 goes beyond this optimization by computing these transforms based only on ${}^{o}\text{supp}(f)$, where $o \in \{\vee, \wedge\}$. Nevertheless, this second approach requires the direct computation of ${}^{o}\text{supp}(f)$, which has an upper bound complexity of $O(|\text{supp}(f)|.|{}^{o}\text{supp}(f)|)$, which may be more than $O(|{}^{\vee}\mathcal{I}(P)|.|P|)$ if $|\text{supp}(f)| \gg |{}^{\vee}\mathcal{I}(P)|$.

3.1 Preliminary results

In this subsection, we provide some propositions that are useful for proving our main results (presented in the next subsection).

Lemma 3.1.1 (*Safe join*). *Let us consider a finite distributive lattice L. For all join-irreducible element $i \in {}^{\vee}\mathcal{I}(L)$ and for all elements $x, y \in L$ that are not greater than or equal to i, i.e. $i \not\leq x$ and $i \not\leq y$, we have that their join is not greater or equal to i either, i.e. $i \not\leq x \vee y$.*

Proof. By definition of a join-irreducible element, we know that $\forall i \in {}^{\vee}\mathcal{I}(L)$ and for all $a, b \in L$, if $a < i$ and $b < i$, then $a \vee b < i$. Moreover, for all $x, y \in L$ such that $i \not\leq x$ and $i \not\leq y$, we have equivalently $i \wedge x < i$ and $i \wedge y < i$. Thus, we get that $(i \wedge x) \vee (i \wedge y) < i$. Since L satisfies the distributive law Eq. (3), this implies that $(i \wedge x) \vee (i \wedge y) = i \wedge (x \vee y) < i$, which means that $i \not\leq x \vee y$. ■

Lemma 3.1.2 (*Safe meet*)**.** *Let us consider a finite distributive lattice L. For all meet-irreducible element $i \in {}^{\wedge}\mathcal{I}(L)$ and for all elements $x, y \in L$ that are not less than or equal to i, i.e. $i \not\geq x$ and $i \not\geq y$, we have that their meet is not less or equal to i either, i.e. $i \not\geq x \wedge y$.*

Proof. By definition of a meet-irreducible element, we know that $\forall i \in {}^{\wedge}\mathcal{I}(L)$ and for all $a, b \in L$, if $a > i$ and $b > i$, then $a \wedge b > i$. Moreover, for all $x, y \in L$ such that $i \not\geq x$ and $i \not\geq y$, we have equivalently $i \vee x > i$ and $i \vee y > i$. Thus, we get that $(i \vee x) \wedge (i \vee y) > i$. Since L satisfies the distributive law Eq. (4), this implies that $(i \vee x) \wedge (i \vee y) = i \vee (x \wedge y) < i$, which means that $i \not\geq x \vee y$. ■

Proposition 3.1.1 (*Iota elements of a subset of P*)**.** For any subset $S \subseteq P$, we note $\iota(S)$ the set containing the join-irreducible elements of the smallest sublattice L_S of P containing S, i.e. $\iota(S) = {}^{\vee}\mathcal{I}(L_S)$. These so-called *iota elements of S* can be obtained through the following equality:

$$\iota(S) = \left\{ \bigwedge i^{\uparrow S} \;/\; i \in {}^{\vee}\mathcal{I}(P),\ i^{\uparrow S} \neq \emptyset \right\},$$

where $i^{\uparrow S}$ is the upper closure of i in S, i.e. $\{s \in S \;/\; i \leq s\}$.

Proof. See Appendix A.1. ■

Proposition 3.1.2 (*Dual iota elements of a subset of P*)**.** Similarly, for any subset $S \subseteq P$, we note $\overline{\iota}(S)$ the set containing the meet-irreducible elements of the smallest sublattice L_S of P containing S, i.e. $\overline{\iota}(S) = {}^{\wedge}\mathcal{I}(L_S)$. These so-called *dual iota elements of S* can be obtained through the following equality:

$$\overline{\iota}(S) = \left\{ \bigvee i^{\downarrow S} \;/\; i \in {}^{\wedge}\mathcal{I}(P),\ i^{\downarrow S} \neq \emptyset \right\},$$

where $i^{\downarrow S}$ is the lower closure of i in S, i.e. $\{s \in S \;/\; i \geq s\}$.

Proof. Analog to the proof of Proposition 3.1.1, exploiting the dual definition of the distributive law in a lattice, i.e. Eq. (4). ■

Proposition 3.1.3 (*Lattice support*)**.** The smallest sublattice of P containing both ${}^{\wedge}\mathrm{supp}(f)$ and ${}^{\vee}\mathrm{supp}(f)$, noted ${}^{\mathcal{L}}\mathrm{supp}(f)$, can be defined as:

$$\begin{aligned} {}^{\mathcal{L}}\mathrm{supp}(f) &= \left\{ \bigvee X \;/\; X \subseteq \iota(\mathrm{supp}(f)),\ X \neq \emptyset \right\} \cup \left\{ \bigwedge \mathrm{supp}(f) \right\} \\ &= \left\{ \bigwedge X \;/\; X \subseteq \overline{\iota}(\mathrm{supp}(f)),\ X \neq \emptyset \right\} \cup \left\{ \bigvee \mathrm{supp}(f) \right\}. \end{aligned}$$

More specifically, ${}^{\vee}\mathrm{supp}(f)$ is contained in the upper closure $\mathrm{supp}(f)^{\uparrow^{\mathcal{L}}\mathrm{supp}(f)}$ of $\mathrm{supp}(f)$ in ${}^{\mathcal{L}}\mathrm{supp}(f)$:

$$\mathrm{supp}(f)^{\uparrow^{\mathcal{L}}\mathrm{supp}(f)} = \{x \in {}^{\mathcal{L}}\mathrm{supp}(f) \;/\; \exists s \in \mathrm{supp}(f),\ s \leq x\},$$

and ${}^{\wedge}\mathrm{supp}(f)$ is contained in the lower closure $\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$ of $\mathrm{supp}(f)$ in ${}^{\mathcal{L}}\mathrm{supp}(f)$:

$$\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)} = \{x \in {}^{\mathcal{L}}\mathrm{supp}(f) \;/\; \exists s \in \mathrm{supp}(f),\ s \geq x\}.$$

These sets can be computed in less than respectively $O(|\iota(\mathrm{supp}(f))|.|\mathrm{supp}(f)^{\uparrow^{\mathcal{L}}\mathrm{supp}(f)}|)$ and $O(|\iota(\mathrm{supp}(f))|.|\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}|)$, which is at most $O(|{}^{\vee}\mathcal{I}(P)|.|P|)$.

Proof. The proof is immediate here, considering Proposition 3.1.1 and its proof, as well as Proposition 3.1.2. In addition, since ${}^{\wedge}\mathrm{supp}(f)$ only contains the meet of elements of $\mathrm{supp}(f)$, all element of ${}^{\wedge}\mathrm{supp}(f)$ is less than at least one element of $\mathrm{supp}(f)$. Similarly, since ${}^{\vee}\mathrm{supp}(f)$ only contains the join of elements of $\mathrm{supp}(f)$, all element of ${}^{\vee}\mathrm{supp}(f)$ is greater than at least one element of $\mathrm{supp}(f)$. Hence $\mathrm{supp}(f)^{\uparrow^{\mathcal{L}}\mathrm{supp}(f)}$ and $\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$. ■

As pointed out in [17], a special ordering of the join-irreducible elements of a lattice when using the Fast Zeta Transform [15] leads to the optimal computation of its zeta and Möbius transforms. Here, we use this ordering to build our EMT for finite distributive lattices in a way similar to [17] but without the need to check the equality of the decompositions into the first j join-irreducible elements at each step.

Corollary 3.1.1 (*Join-irreducible ordering*)**.** *Let us consider a finite distributive lattice* $(L, \leq)$ *and let its join-irreducible elements* ${}^{\vee}\mathcal{I}(L)$ *be ordered such that* $\forall i_k, i_l \in {}^{\vee}\mathcal{I}(L)$, $k < l \Rightarrow i_k \not\geq i_l$. *If* L *is a graded lattice (i.e. a lattice equipped with a rank function* $\rho : L \to \mathbb{N}$*), then* $\rho(i_1) \leq \rho(i_2) \leq \cdots \leq \rho(i_{|{}^{\vee}\mathcal{I}(L)|})$ *implies this ordering. We note* ${}^{\vee}\mathcal{I}(L)_k = \{i_1, \ldots, i_{k-1},\ i_k\}$.

For all element $i_k \in {}^{\vee}\mathcal{I}(L)$*, we have* $i_k \not\leq \bigvee {}^{\vee}\mathcal{I}(L)_{k-1}$.

Proof. Since the join-irreducible elements are ordered such that $\forall i_k, i_l \in {}^{\vee}\mathcal{I}(L)$, $k < l \Rightarrow i_k \not\geq i_l$, it is trivial to see that for any $i_l \in {}^{\vee}\mathcal{I}(L)$ and $i_k \in {}^{\vee}\mathcal{I}(L)_{l-1}$, we have $i_k \not\geq i_l$. Then, using Lemma 3.1.1 by recurrence, it is easy to get that $i_l \not\leq \bigvee {}^{\vee}\mathcal{I}(L)_{l-1}$. ■

Example 3.1.1. For example, in DST we work with $P = 2^{\Omega}$, in which the rank function is the cardinality, i.e. for all $A \in P$, $\rho(A) = |A|$. So, with $(L, \subseteq)$ a subset lattice of $(2^{\Omega}, \subseteq)$, the ordering required in Corollary 3.1.1 simply translates to sorting the join-irreducible elements of L from the smallest set to the largest set. Thus, sorting ${}^{\vee}\mathcal{I}(L) = \{i_1, i_2, \cdots, i_n\}$ such that $|i_1| \leq |i_2| \leq \cdots \leq |i_n|$, Corollary 3.1.1 tells us that for any $k \in [\![2, n]\!]$, we have $i_k \not\subseteq \bigcup {}^{\vee}\mathcal{I}(L)_{k-1}$, where ${}^{\vee}\mathcal{I}(L)_k = \{i_1, \ldots, i_{k-1},\ i_k\}$.

Corollary 3.1.2 (*Meet-irreducible ordering*)**.** *Let us consider a finite distributive lattice* $(L, \leq)$ *and let its meet-irreducible elements* ${}^{\wedge}\mathcal{I}(L)$ *be ordered such that* $\forall i_k, i_l \in {}^{\vee}\mathcal{I}(L)$, $k < l \Rightarrow i_k \not\leq i_l$. *If* L *is a graded lattice (i.e. a lattice equipped with a rank function* $\rho : L \to \mathbb{N}$*), then* $\rho(i_1) \geq \rho(i_2) \geq \cdots \geq \rho(i_{|{}^{\wedge}\mathcal{I}(L)|})$ *implies this ordering. We note* ${}^{\wedge}\mathcal{I}(L)_k = \{i_1, \ldots, i_{k-1},\ i_k\}$.

For all element $i_k \in {}^{\wedge}\mathcal{I}(L)$*, we have* $i_k \not\geq \bigwedge {}^{\wedge}\mathcal{I}(L)_{k-1}$.

Proof. Since the meet-irreducible elements are ordered such that $\forall i_k, i_l \in {}^{\wedge}\mathcal{I}(L)$, $k < l \Rightarrow i_k \not\leq i_l$, it is trivial to see that for any $i_l \in {}^{\wedge}\mathcal{I}(L)$ and $i_k \in {}^{\wedge}\mathcal{I}(L)_{l-1}$, we have $i_k \not\leq i_l$. Then, using Lemma 3.1.2 by recurrence, it is easy to get that $i_l \not\geq \bigwedge {}^{\wedge}\mathcal{I}(L)_{l-1}$. ■

Example 3.1.2. Taking back Example 3.1.1, the ordering required in this Corollary 3.1.2 simply translates to sorting the meet-irreducible elements of L from the largest set to the smallest set. Thus, sorting ${}^{\wedge}\mathcal{I}(L) = \{i_1, i_2, \cdots, i_n\}$ such that $|i_1| \geq |i_2| \geq \cdots \geq |i_n|$, Corollary 3.1.2 tells us that for any $k \in [\![2, n]\!]$, we have $i_k \not\supseteq \bigcap {}^{\wedge}\mathcal{I}(L)_{k-1}$, where ${}^{\wedge}\mathcal{I}(L)_k = \{i_1, \ldots, i_{k-1},\ i_k\}$.

3.2 Main results

In this subsection, we present our two types of transformations computing the zeta and Möbius transforms of a function $f : P \to \mathbb{R}$. The first one corresponds to Theorem 3.2.1 and its corollaries and is based on a sublattice $L \subseteq P$, where L contains the lattice support of f, i.e. ${}^{\mathcal{L}}\text{supp}(f) \subseteq L$. The second one corresponds to Theorem 3.2.2 and its corollary and is based on any lower subsemilattice of P containing ${}^{\wedge}\text{supp}(f)$. Its dual is presented in Corollary 3.2.7 and its corollary and is based on any upper subsemilattice of P containing ${}^{\vee}\text{supp}(f)$.

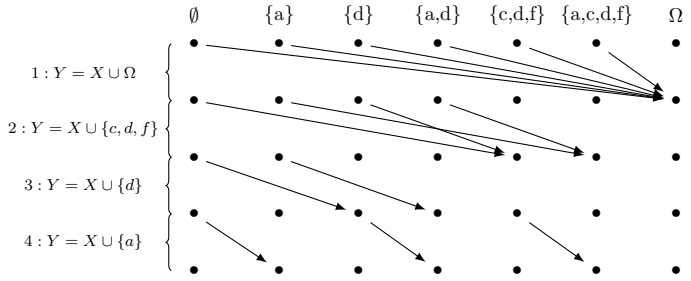

Figure 3: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,4]\!]}$, where $H_k = (L, E_k)$ and $E_k = \{(X, Y) \in L^2 \ / \ Y = X \cup i_{5-k}\}$ and $L = \{\emptyset, \{a\}, \{d\}, \{a, d\}, \{c, d, f\}, \{a, c, d, f\}, \Omega\}$ and $\Omega = \{a, b, c, d, e, f\}$ and $(i_k)_{k\in[\![1,4]\!]} = (\{a\}, \{d\}, \{c, d, f\}, \Omega)$. This sequence computes the same zeta transformations as $G_\subset = (L, E_\subset)$, where $E_\subset = \{(X, Y) \in L^2 \ / \ X \subset Y\}$.

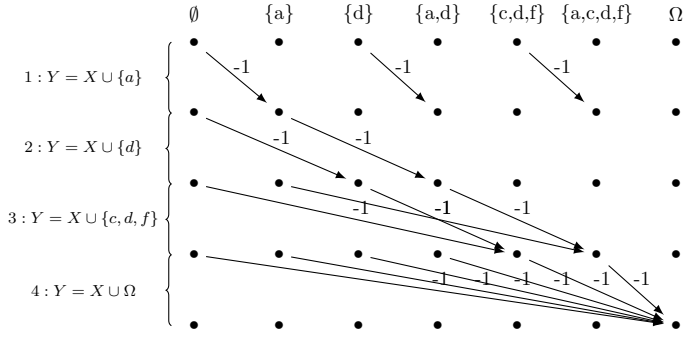

Figure 4: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,4]\!]}$, where $H_k = (L, E_k)$ and $E_k = \{(X, Y) \in L^2 \ / \ Y = X \cup i_k\}$ and $L = \{\emptyset, \{a\}, \{d\}, \{a, d\}, \{c, d, f\}, \{a, c, d, f\}, \Omega\}$ and $\Omega = \{a, b, c, d, e, f\}$ and $(i_k)_{k\in[\![1,4]\!]} = (\{a\}, \{d\}, \{c, d, f\}, \Omega)$. This sequence computes the same Möbius transformations as $G_\subset = (L, E_\subset)$, where $E_\subset = \{(X, Y) \in L^2 \ / \ X \subset Y\}$. The "-1" labels emphasize the intended use of the operator $-$ with this sequence.

Theorem 3.2.1 (*Efficient Möbius Transformation in a distributive lattice*)**.** *Let us consider a finite distributive lattice L (such as ${}^{\mathcal{L}}supp(f)$) and let its join-irreducible elements ${}^{\vee}\mathcal{I}(L) = \{i_1, i_2, \cdots, i_n\}$ be ordered such that $\forall i_k, i_l \in {}^{\vee}\mathcal{I}(L)$, $k < l \Rightarrow i_k \not\geq i_l$.*

Consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (L, E_k)$ and:

$$E_k = \left\{(x, y) \in L^2 \ / \ y = x \vee i_{n+1-k}\right\}.$$

This sequence computes the same zeta transformations as $G_< = (L, E_<)$, where $E_< = \{(X, Y) \in L^2 \ / \ X < Y\}$. This sequence is illustrated in Fig. 3. The execution of any transformation based on this sequence is $O(n.|L|)$ in time and $O(|L|)$ in space.

Proof. See Appendix A.2. ■

Corollary 3.2.1. *It can be shown similarly that the sequence $(H_k)_{k\in[\![1,n]\!]}$ computes the same zeta transforms as the sequence of Theorem 3.2.1, where $H_k = (L, E_k)$ and:*

$$E_k = \left\{(x, y) \in L^2 \ / \ x = y \wedge \overline{i}_k\right\}.$$

with the meet-irreducible elements ${}^{\wedge}\mathcal{I}(L) = \{\overline{i}_1, \overline{i}_2, \cdots, \overline{i}_n\}$ ordered such that $\forall \overline{i}_k, \overline{i}_l \in {}^{\wedge}\mathcal{I}(L)$,

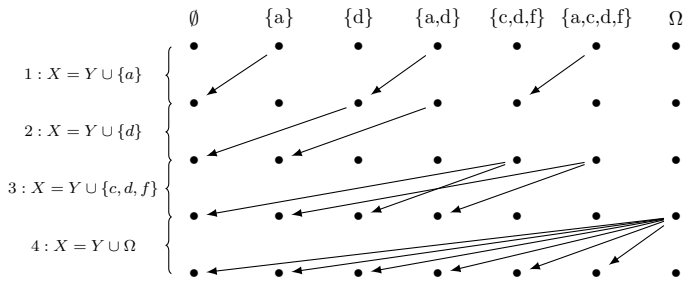

Figure 5: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,4]\!]}$, where $H_k = (L, E_k)$ and $E_k = \{(X,Y) \in L^2 \ / \ X = Y \cup i_k\}$ and $L = \{\emptyset, \{a\}, \{d\}, \{a,d\}, \{c,d,f\}, \{a,c,d,f\}, \Omega\}$ and $\Omega = \{a,b,c,d,e,f\}$ and $(i_k)_{k\in[\![1,4]\!]} = (\{a\}, \{d\}, \{c,d,f\}, \Omega)$. This sequence computes the same zeta transformations as $G_\supset = (L, E_\supset)$, where $E_\supset = \{(X,Y) \in L^2 \ / \ X \supset Y\}$.

$k < l \Rightarrow \overline{i}_k \not\leq \overline{i}_l$, i.e. in reverse order compared to the join-irreducible elements of Theorem 3.2.1.

Corollary 3.2.2. *Consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (L, E_k)$ and:*

$$E_k = \left\{(x,y) \in L^2 \ / \ y = x \vee i_k\right\}.$$

This sequence computes the same Möbius transformations as $G_< = (L, E_<)$, where $E_< = \{(X,Y) \in L^2 \ / \ X < Y\}$. This sequence is illustrated in Fig. 4 and leads to the same complexities as the one presented in Theorem 3.2.1.

Proof. For every arrow $(x,y) \in E_k$, if $i_k \not\leq x$, then there is an arrow such that $y = x \vee i_k \neq x$. However, there can be no arrow $(w,x) \in E_k$ since x cannot be equal to $w \vee i_k$. Otherwise, if $i_k \leq x$, then $y = x \vee i_k = x$, i.e. (x,y) is an identity arrow. Thus, for every arrow $(x,y) \in E_k \backslash I_P$, there is no arrow (w,x) in $E_k \backslash I_P$, meaning that Theorem 2.4.1 is satisfied. The sequence $(H_{n-k+1})_{k\in[\![1,n]\!]}$ computes the same Möbius transformations as $G_<$. ■

Corollary 3.2.3. *Again, it can be shown similarly that the sequence $(H_k)_{k\in[\![1,n]\!]}$ computes the same Möbius transforms as the sequence of Corollary 3.2.2, where $H_k = (L, E_k)$ and:*

$$E_k = \left\{(x,y) \in L^2 \ / \ x = y \wedge \overline{i}_{n+1-k}\right\}.$$

with the meet-irreducible elements $^\wedge\mathcal{I}(L) = \{\overline{i}_1, \overline{i}_2, \cdots, \overline{i}_n\}$ ordered such that $\forall \overline{i}_k, \overline{i}_l \in {}^\wedge\mathcal{I}(L)$, $k < l \Rightarrow \overline{i}_k \not\leq \overline{i}_l$, i.e. in reverse order compared to the join-irreducible elements of Corollary 3.2.2.

Corollary 3.2.4. *Dually, consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (L, E_k)$ and:*

$$E_k = \left\{(x,y) \in L^2 \ / \ x = y \vee i_k\right\}.$$

This sequence computes the same zeta transformations as $G_> = (L, E_>)$, where $E_> = \{(X,Y) \in L^2 \ / \ X > Y\}$. This sequence is illustrated in Fig. 5 and leads to the same complexities as its dual.

Corollary 3.2.5. *Consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (L, E_k)$ and:*

$$E_k = \left\{(x,y) \in L^2 \ / \ x = y \vee i_{n+1-k}\right\}.$$

This sequence computes the same Möbius transformations as $G_> = (L, E_>)$, where $E_> = \{(X,Y) \in L^2 \ / \ X > Y\}$. This sequence is illustrated in Fig. 6 and leads to the same complexities as the one presented in Theorem 3.2.1.

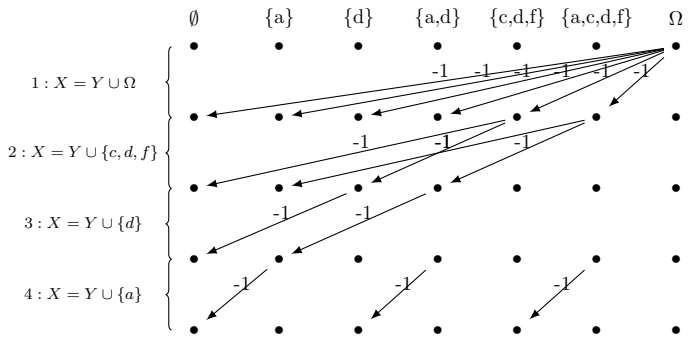

Figure 6: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,4]\!]}$, where $H_k = (L, E_k)$ and $E_k = \{(X,Y) \in L^2 \ / \ X = Y \cup i_{5-k}\}$ and $L = \{\emptyset, \{a\}, \{d\}, \{a,d\}, \{c,d,f\}, \{a,c,d,f\}, \Omega\}$ and $\Omega = \{a,b,c,d,e,f\}$ and $(i_k)_{k\in[\![1,4]\!]} = (\{a\}, \{d\}, \{c,d,f\}, \Omega)$. This sequence computes the same Möbius transformations as $G_\supset = (L, E_\supset)$, where $E_\supset = \{(X,Y) \in L^2 \ / \ X \supset Y\}$. The "-1" labels emphasize the intended use of the operator $-$ with this sequence.

The procedures exploiting the sequences of graphs described in Theorem 3.2.1 and its corollaries to compute the zeta and Möbius transforms of a function f on P is always less than $O(|^\vee\mathcal{I}(P)|.|P|)$. Their upper bound complexity for the distributive lattice $L = {}^{\mathcal{L}}\text{supp}(f)$ is $O(|^\vee\mathcal{I}(L)|.|L|)$, which is actually the optimal one for a lattice.

Yet, we can reduce this complexity even further if we have ${}^\wedge\text{supp}(f)$ or ${}^\vee\text{supp}(f)$. This is the motivation behind the sequence proposed in the following Theorem 3.2.2. As a matter of fact, [15] proposed a meta-procedure producing an algorithm that computes the zeta and Möbius transforms in an arbitrary intersection-closed family F of sets of 2^Ω with a circuit of size $O(|\Omega|.|F|)$. However, this meta-procedure is $O(|\Omega|.2^{|\Omega|})$. Here, Theorem 3.2.2 provides a sequence of graphs that leads to procedures that directly compute the zeta and Möbius transforms in $O(|\Omega|.|F|.\epsilon)$, where ϵ can be as low as 1. Besides, our method is far more general since it applies to any meet-closed or join-closed subset of a finite distributive lattice. The intuition behind it is that we can do the same computations as in Theorem 3.2.1, even in a subsemilattice, simply by bridging gaps, from the smallest gap to the biggest to make sure that all nodes are visited.

Theorem 3.2.2 (*Efficient Möbius Transformation in a join-closed or meet-closed subset of P*)**.** *Let us consider a meet-closed subset M of P (such as ${}^\wedge supp(f)$). Also, let the iota elements $\iota(M) = \{i_1, i_2, \cdots, i_n\}$ be ordered such that $\forall i_k, i_l \in \iota(M)$, $k < l \Rightarrow i_k \not\geq i_l$.*

Consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (M, E_k)$ and:

$$E_k = \left\{(x,y) \in M^2 \ / \ x = \bigwedge (y \vee i_k)^{\uparrow M} \text{ and } x \leq y \vee \bigvee \iota(M)_k \right\},$$

where $\iota(M)_k = \{i_1, i_2, \ldots, i_k\}$. This sequence computes the same zeta transformations as $G_> = (M, E_>)$, where $E_> = \{(X,Y) \in M^2 \ / \ X > Y\}$. This sequence is illustrated in Fig. 7. The execution of any transformation based on this sequence is at most $O(|\iota(M)|.|M|)$ in space and $O(|\iota(M)|.|M|.\epsilon)$ in time, where ϵ represents the average number of operations required to "bridge a gap", i.e. to find the minimum of $(y \vee i_k)^{\uparrow M}$.

Proof. See Appendix A.3 ■

Remark. This number ϵ is hard to evaluate beforehand since it depends on both the number of "gaps to bridge" and the average number of elements that can be greater than some element.

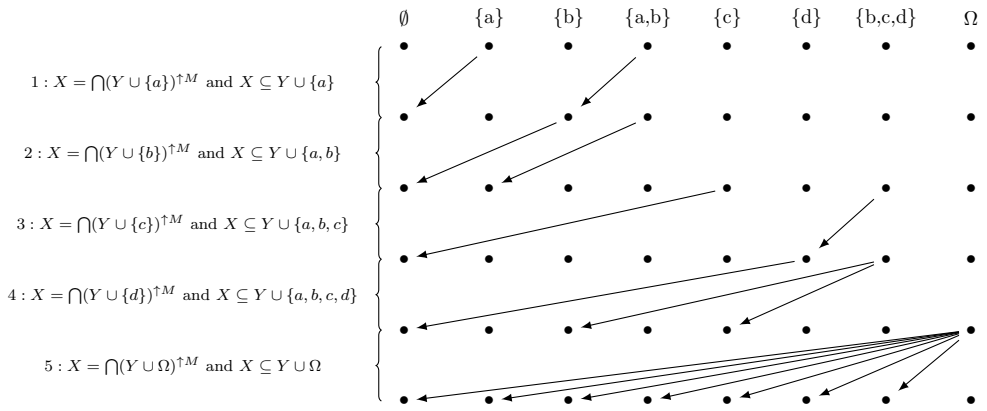

Figure 7: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,5]\!]}$, where $H_k = (M, E_k)$ and $E_k = \left\{(X, Y) \in M^2 \ / \ X = \bigcap(Y \cup i_k)^{\uparrow M} \text{ and } X \subseteq Y \cup \bigcup \iota(M)_k\right\}$ and $\iota(M)_k = \{i_1, i_2, \ldots, i_k\}$ and $M = \{\emptyset, \{a\}, \{b\}, \{a, b\}, \{c\}, \{d\}, \{b, c, d\}, \Omega\}$ and $\Omega = \{a, b, c, d, e, f\}$ and $(i_k)_{k\in[\![1,5]\!]} = (\{a\}, \{b\}, \{c\}, \{d\}, \Omega)$. This sequence computes the same zeta transformations as $G_\supset = (M, E_\supset)$, where $E_\supset = \{(X, Y) \in M^2 \ / \ X \supset Y\}$. Actually, since there is no order between any two iota elements of $\iota(M)\backslash\{\Omega\}$ in this example, the order chosen here is arbitrary. Any order would compute the same zeta transformations as $G_\supset$, as long as Ω is the last iota element to consider.

If there is no gap, or if there is only one greater element, this ϵ can be as low as 1. Moreover, the choice of data structures may greatly reduce this ϵ. For instance, if $P = 2^\Omega$, then dynamic binary trees may be employed to avoid the consideration of elements that cannot be less than some already found element in $(y \cup i_k)^{\uparrow M}$, by cutting some branches.

Corollary 3.2.6. *Consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (M, E_k)$ and:*

$$E_k = \left\{(x, y) \in M^2 \ / \ x = \bigwedge(y \vee i_{n+1-k})^{\uparrow M} \text{ and } x \leq y \vee \bigvee \iota(M)_{n+1-k}\right\}.$$

This sequence computes the same Möbius transformations as $G_> = (M, E_>)$, where $E_> = \{(X, Y) \in M^2 \ / \ X > Y\}$. This sequence is illustrated in Fig. 8 and leads to the same complexities as the one presented in Theorem 3.2.2.

Proof. Analog to the proof of Corollary 3.2.2. ■

Corollary 3.2.7. *Dually, let us consider a join-closed subset J of P (such as ${}^\vee supp(f)$). Also, let the dual iota elements $\overline{\iota}(J) = \{\overline{i}_1, \overline{i}_2, \cdots, \overline{i}_n\}$ be ordered such that $\forall \overline{i}_k, \overline{i}_l \in \overline{\iota}(J)$, $k < l \Rightarrow \overline{i}_k \not\leq \overline{i}_l$, i.e. in reverse order compared to the iota elements of Theorem 3.2.2.*

In the direct line of Corollary 3.2.1, consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (J, E_k)$ and:

$$E_k = \left\{(x, y) \in J^2 \ / \ x = \bigvee(y \wedge \overline{i}_k)^{\downarrow J} \text{ and } x \geq y \wedge \bigwedge \overline{\iota}(J)_k\right\},$$

where $\overline{\iota}(J)_k = \{\overline{i}_1, \overline{i}_2, \ldots, \overline{i}_k\}$. This sequence computes the same zeta transformations as $G_< = (J, E_<)$, where $E_< = \{(X, Y) \in J^2 \ / \ X < Y\}$. This sequence is illustrated in Fig. 9. The execution of any transformation based on this sequence is at most $O(|\overline{\iota}(J)|.|J|)$ in space and $O(|\overline{\iota}(J)|.|J|.\epsilon)$ in time, where ϵ represents the average number of operations required to "bridge a gap", i.e. to find the maximum of $(y \wedge \overline{i}_k)^{\downarrow M}$.

Corollary 3.2.8. *Finally, in the direct line of Corollary 3.2.3, consider the sequence $(H_k)_{k\in[\![1,n]\!]}$,*

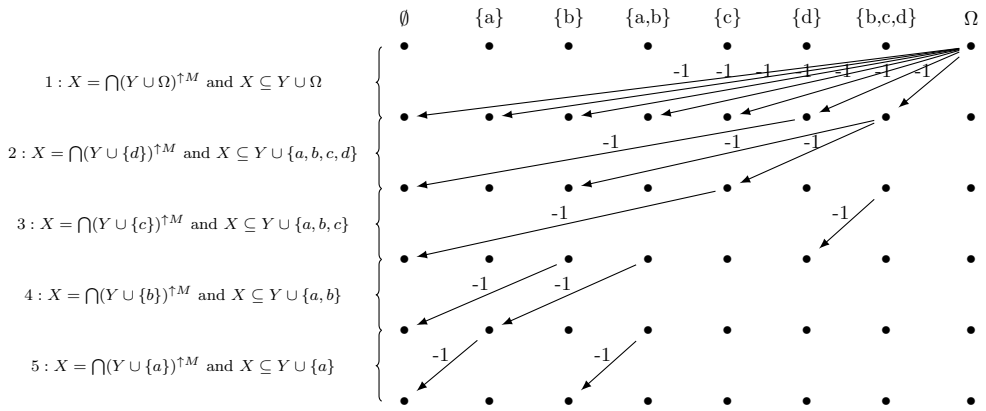

Figure 8: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,5]\!]}$, where $H_k = (M, E_k)$ and $E_k = \left\{(X, Y) \in M^2 \ / \ X = \bigcap(Y \cup i_{6-k})^{\uparrow M} \text{ and } X \subseteq Y \cup \bigcup \iota(M)_{6-k}\right\}$ and $\iota(M)_k = \{i_1, i_2, \ldots, i_k\}$ and $M = \{\emptyset, \{a\}, \{b\}, \{a, b\}, \{c\}, \{d\}, \{b, c, d\}, \Omega\}$ and $\Omega = \{a, b, c, d, e, f\}$ and $(i_k)_{k\in[\![1,5]\!]} = (\{a\}, \{b\}, \{c\}, \{d\}, \Omega)$. This sequence computes the same Möbius transformations as $G_\supset = (M, E_\supset)$, where $E_\supset = \{(X, Y) \in M^2 \ / \ X \supset Y\}$. The "-1" labels emphasize the intended use of the operator $-$ with this sequence. Actually, since there is no order between any two iota elements of $\iota(M)\backslash\{\Omega\}$ in this example, the order chosen in Fig. 7 is arbitrary. Thus, any order here would compute the same Möbius transformations as $G_\supset$, as long as Ω is the first iota element to consider.

where $H_k = (J, E_k)$ and:

$$E_k = \left\{(x, y) \in J^2 \ / \ x = \bigvee (y \wedge \overline{i}_{n+1-k})^{\downarrow J} \text{ and } x \geq y \wedge \bigwedge \overline{\iota}(J)_{n+1-k}\right\}.$$

This sequence computes the same Möbius transformations as $G_< = (J, E_<)$, where $E_< = \{(X, Y) \in J^2 \ / \ X < Y\}$. This sequence is illustrated in Fig. 10 and leads to the same complexities as the one presented in Corollary 3.2.7.

4 Discussions

4.1 General usage

If $|\text{supp}(f)|$ is of same order of magnitude as $|^\vee\mathcal{I}(P)|$ or lower, then we can directly compute the focal points $^\wedge\text{supp}(f)$ or $^\vee\text{supp}(f)$. Next, with $^\wedge\text{supp}(f)$, we can compute Efficient Möbius Transformations based on Theorem 3.2.2 to get the zeta or Möbius transform of any function f in $(P, \geq)$.

Let us take the sequence $(H_k)_{k\in[\![1,n]\!]}$ from Theorem 3.2.2, where $H_k = (^\wedge\text{supp}(f), E_k)$ and:

$$E_k = \left\{(x, y) \in {}^\wedge\text{supp}(f)^2 \ / \ x = \bigwedge (y \vee i_k)^{\uparrow^{\wedge}\text{supp}(f)} \text{ and } x \leq y \vee \bigvee \iota(^\wedge\text{supp}(f))_k\right\},$$

where $\iota(^\wedge\text{supp}(f))_k = \{i_1, i_2, \ldots, i_k\}$ such that $\forall i_k, i_l \in \iota(^\wedge\text{supp}(f))$, $k < l \Rightarrow i_k \not\geq i_l$.

Example 4.1.1. Consider the mass function m and the commonality function q from Example 2.1.3. The transformation $((H_k)_{k\in[\![1,n]\!]}, m, +)$, where $^\wedge\text{supp}(f) = {}^\wedge\text{supp}(m)$, computes the commonality function q.

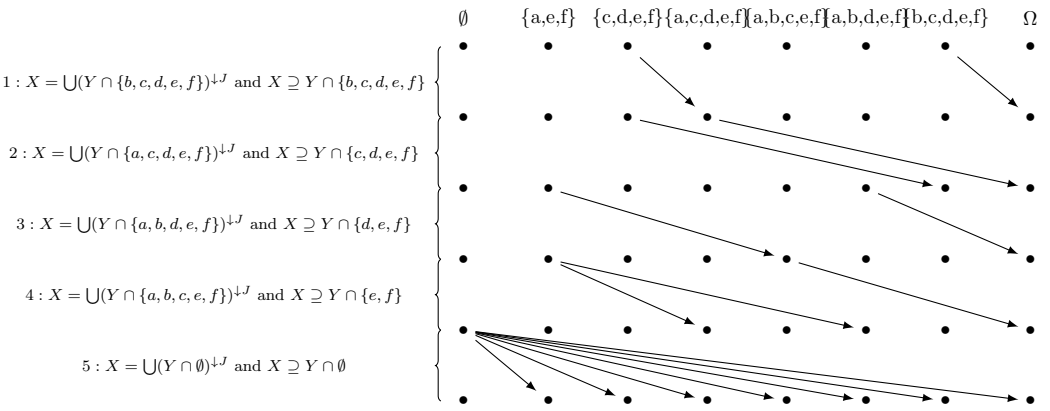

Figure 9: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,5]\!]}$, where $H_k = (J, E_k)$ and $E_k = \left\{(X,Y) \in J^2 \ / \ X = \bigcup(Y \cap \overline{i}_k)^{\downarrow J} \text{ and } X \supseteq Y \cap \bigcap \overline{\iota}(J)_k\right\}$ and $\overline{\iota}(J)_k = \{\overline{i}_1, \overline{i}_2, \ldots, \overline{i}_k\}$ and $J = \{\emptyset, \{a,e,f\}, \{c,d,e,f\}, \{a,c,d,e,f\}, \{a,b,c,e,f\}, \{a,b,d,e,f\}, \{b,c,d,e,f\}, \Omega\}$ and $\Omega = \{a,b,c,d,e,f\}$ and $(\overline{i}_k)_{k\in[\![1,5]\!]} = (\{b,c,d,e,f\}, \{a,c,d,e,f\}, \{a,b,d,e,f\}, \{a,b,c,e,f\}, \emptyset)$. This sequence computes the same zeta transformations as $G_\subset = (J, E_\subset)$, where $E_\subset = \{(X,Y) \in J^2 \ / \ X \subset Y\}$. Actually, since there is no order between any two dual iota elements of $\overline{\iota}(J)\backslash\{\emptyset\}$ in this example, the order chosen here is arbitrary. Any order would compute the same zeta transformations as $G_\subset$, as long as $\emptyset$ is the last dual iota element to consider.

Example 4.1.2. Consider the mass function m and the commonality function q from Example 2.2.3. The transformation $((H_{n+1-k})_{k\in[\![1,n]\!]}, q, -)$, where ${}^\wedge\mathrm{supp}(f) = {}^\wedge\mathrm{supp}(m)$, computes the mass function m.

Example 4.1.3. Consider the conjunctive weight function w and the commonality function q from Example 2.1.4. The transformation $((H_k)_{k\in[\![1,n]\!]}, w^{-1}, \times)$, where ${}^\wedge\mathrm{supp}(f) = {}^\wedge\mathrm{supp}(w-1)$, computes the commonality function q.

Example 4.1.4. Consider the conjunctive weight function w and the commonality function q from Example 2.2.4. The transformation $((H_{n+1-k})_{k\in[\![1,n]\!]}, w^{-1}, /)$, where ${}^\wedge\mathrm{supp}(f) = {}^\wedge\mathrm{supp}(w-1)$, computes the conjunctive weight function w.

Let us now take the sequence $(H_k)_{k\in[\![1,n]\!]}$ from Corollary 3.2.7, where $H_k = ({}^\vee\mathrm{supp}(f), E_k)$ and:

$$E_k = \left\{(x,y) \in {}^\vee\mathrm{supp}(f)^2 \ / \ x = \bigvee(y \wedge \overline{i}_k)^{\downarrow {}^\vee\mathrm{supp}(f)} \text{ and } x \geq y \wedge \bigwedge \overline{\iota}({}^\vee\mathrm{supp}(f))_k\right\},$$

where $\overline{\iota}({}^\vee\mathrm{supp}(f))_k = \{\overline{i}_1, \overline{i}_2, \ldots, \overline{i}_k\}$ and such that $\forall \overline{i}_k, \overline{i}_l \in \overline{\iota}({}^\vee\mathrm{supp}(f))$, $k < l \Rightarrow \overline{i}_k \not\leq \overline{i}_l$.

Example 4.1.5. Consider the mass function m and the implicability function b from Example 2.1.1. The transformation $((H_k)_{k\in[\![1,n]\!]}, m, +)$, where ${}^\vee\mathrm{supp}(f) = {}^\vee\mathrm{supp}(m)$, computes the implicability function b.

Example 4.1.6. Consider the mass function m and the implicability function b from Example 2.2.1. The transformation $((H_{n+1-k})_{k\in[\![1,n]\!]}, b, -)$, where ${}^\vee\mathrm{supp}(f) = {}^\vee\mathrm{supp}(m)$, computes the mass function m.

Example 4.1.7. Consider the disjunctive weight function v and the implicability function b from Example 2.1.2. The transformation $((H_k)_{k\in[\![1,n]\!]}, v^{-1}, \times)$, where ${}^\vee\mathrm{supp}(f) = {}^\vee\mathrm{supp}(v-1)$, computes the implicability function b.

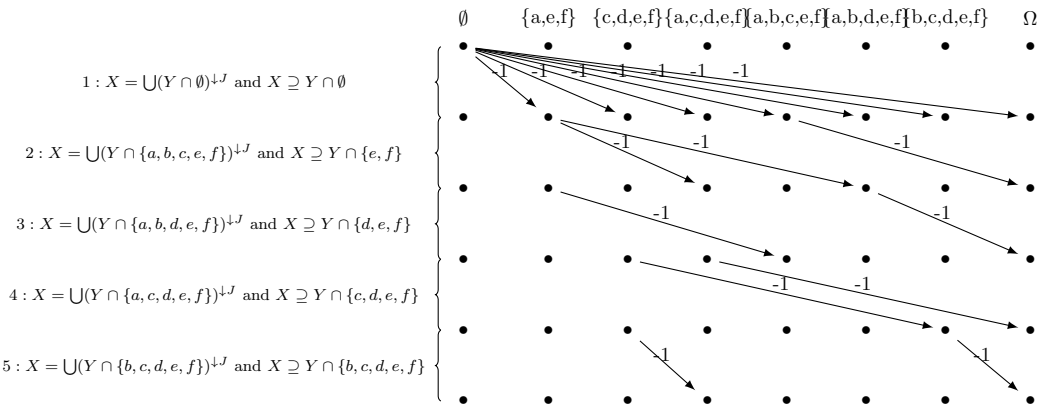

Figure 10: Illustration representing the paths generated by the arrows contained in the sequence $(H_k)_{k\in[\![1,5]\!]}$, where $H_k = (J, E_k)$ and $E_k = \{(X,Y) \in J^2 \ / \ X = \bigcup(Y \cap \overline{i}_{6-k})^{\downarrow J} \text{ and } X \supseteq Y \cap \bigcap \overline{\iota}(J)_{6-k}\}$ and $\overline{\iota}(J)_k = \{\overline{i}_1, \overline{i}_2, \ldots, \overline{i}_k\}$ and $J = \{\emptyset, \{a,e,f\}, \{c,d,e,f\}, \{a,c,d,e,f\}, \{a,b,c,e,f\}, \{a,b,d,e,f\}, \{b,c,d,e,f\}, \Omega\}$ and $\Omega = \{a,b,c,d,e,f\}$ and $(\overline{i}_k)_{k\in[\![1,5]\!]} = (\{b,c,d,e,f\}, \{a,c,d,e,f\}, \{a,b,d,e,f\}, \{a,b,c,e,f\}, \emptyset)$. This sequence computes the same Möbius transformations as $G_\subset = (J, E_\subset)$, where $E_\subset = \{(X,Y) \in J^2 \ / \ X \subset Y\}$. The "-1" labels emphasize the intended use of the operator $-$ with this sequence. Actually, since there is no order between any two dual iota elements of $\overline{\iota}(J)\backslash\{\emptyset\}$ in this example, the order chosen in Fig. 9 is arbitrary. Thus, any order would compute the same Möbius transformations as $G_\subset$, as long as $\emptyset$ is the first dual iota element to consider.

Example 4.1.8. Consider the disjunctive weight function v and the implicability function b from Example 2.2.2. The transformation $((H_{n+1-k})_{k\in[\![1,n]\!]}, v^{-1}, /)$, where ${}^\vee\mathrm{supp}(f) = {}^\vee\mathrm{supp}(v-1)$, computes the disjunctive weight function v.

These transformations can be computed in at most $O(|{}^\vee\mathcal{I}(P)|.|\mathrm{supp}(f)|+|I(\mathrm{supp}(f))|.|{}^o\mathrm{supp}(f)|)$ operations, where $I \in \{\iota, \overline{\iota}\}$ and $o \in \{\wedge, \vee\}$, which is at most $O(|{}^\vee\mathcal{I}(P)|.|P|)$.

Otherwise, if $|\mathrm{supp}(f)| \gg |{}^\vee\mathcal{I}(P)|$, then we can compute $\mathrm{supp}(f)^{\uparrow^{\mathcal{L}}\mathrm{supp}(f)}$ or $\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$ from the lattice support of Proposition 3.1.3, and then compute Efficient Möbius Transformations based on Theorem 3.2.1. Doing so, computing the same transforms can be done in at most $O(|{}^\vee\mathcal{I}(P)|.|\mathrm{supp}(f)| + |I(\mathrm{supp}(f))|.|\mathrm{supp}(f)^{A^{\mathcal{L}}\mathrm{supp}(f)}|)$ operations, where $I \in \{\iota, \overline{\iota}\}$ and $A \in \{\uparrow, \downarrow\}$, which is at most $O(|{}^\vee\mathcal{I}(P)|.|P|)$.

Either way, it is always possible to compute zeta and Möbius transforms in a distributive lattice in less than $O(|{}^\vee\mathcal{I}(P)|.|P|)$ in time and space.

4.2 Dempster-Shafer Theory

So, we can always compute most DST transformations (See Examples 4.1.1 to 4.1.8), wherever the FMT applies, in less than $O(|\Omega|.2^{|\Omega|})$ operations in the general case. The EMT are always more efficient than the FMT.

Moreover, $\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$ can be optimized if $\Omega \in \mathrm{supp}(f)$. Indeed, in this case, we have $\mathrm{supp}(f)^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)} = {}^{\mathcal{L}}\mathrm{supp}(f)$, while there may be a lot less elements in $(\mathrm{supp}(f)\backslash\{\Omega\})^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$. If so, one can equivalently compute the down set $(\mathrm{supp}(f)\backslash\{\Omega\})^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$, execute an EMT with Theorem 3.2.1, and then add the value on Ω to the value on all sets of $(\mathrm{supp}(f)\backslash\{\Omega\})^{\downarrow^{\mathcal{L}}\mathrm{supp}(f)}$.

The same can be done with $(\text{supp}(f)\backslash\{\emptyset\})^{\uparrow^{\mathcal{L}}\text{supp}(f)}$. This trick can be particularly useful in the case of the conjunctive and disjunctive weight function, which require that $\text{supp}(f)$ contains respectively Ω and $\emptyset$.

Also, optimizations built for the FMT, such as the reduction of Ω to the core $\mathcal{C}$ or its optimal coarsened version Ω', are already encoded in the use of the function ι (see Example 4.2.1). On the other hand, optimizations built for the evidence-based approach, such as approximations by reduction of the number of focal sets, i.e. reducing the size of $\text{supp}(f)$, can still greatly enhance the EMT.

Finally, it was proposed in [8] to fuse two mass functions m_1 and m_2 using Dempster's rule by computing the corresponding commonality functions q_1 and q_2 in $O(|\Omega|.2^{|\Omega|})$, then computing $q_{12} = q_1.q_2$ in $O(2^{|\Omega|})$ and finally computing back the fused mass function m_{12} from q_{12} in $O(|\Omega|.2^{|\Omega|})$. Here, we propose to compute the same detour but only on the elements of $\text{supp}(m_{12}) \subseteq (\text{supp}(m_1) \cup \text{supp}(m_2))^{\downarrow^{\mathcal{L}}\text{supp}(f)}$. Indeed, notice that $\text{supp}(m_{12}) \subseteq {}^{\wedge}(\text{supp}(m_1) \cup \text{supp}(m_2))$, which implies that $\text{supp}(m_{12})^{\downarrow^{\mathcal{L}}\text{supp}(f)} \subseteq (\text{supp}(m_1) \cup \text{supp}(m_2))^{\downarrow^{\mathcal{L}}\text{supp}(f)}$. Thus, noting $L = (\text{supp}(m_1) \cup \text{supp}(m_2))^{\downarrow^{\mathcal{L}}\text{supp}(f)}$, we compute the corresponding commonality functions q_1 and q_2 in $O(|\iota(L)|.|L|)$, then compute $q_{12} = q_1.q_2$ in $O(|L|)$ and finally compute back the fused mass function m_{12} from q_{12} in $O(|\iota(L)|.|L|)$, where $\iota(L) = \iota(\text{supp}(m_1) \cup \text{supp}(m_2))$.

Example 4.2.1 (*Coarsening in the consonant case*)**.** Let $\text{supp}(f) = \{F_1,\ F_2,\ \ldots,\ F_K\}$ such that $F_1 \subset F_2 \subset \cdots \subset F_K$. A coarsening Ω' of Ω is a mapping from disjoint groups of elements of Ω to elements of Ω'. The set Ω' can be seen as a partition of Ω. The goal of this coarsening of Ω is to provide a reduced powerset $2^{\Omega'}$. The best coarsening in this example would create as much elements in Ω' as there are elements in $\text{supp}(f)$. Thus, the best coarsening would give us a powerset of size $2^{|\text{supp}(f)|}$.

On the other hand, our iota elements $\iota(\text{supp}(f))$ are the join-irreducible elements of the smallest sublattice of 2^{Ω} containing $\text{supp}(f)$. This lattice is what we called the *lattice support of* f and noted ${}^{\mathcal{L}}\text{supp}(f)$. By definition, we necessarily have $|{}^{\mathcal{L}}\text{supp}(f)| \leq 2^{|\text{supp}(f)|}$. More precisely here, all elements of $\text{supp}(f)$ are both focal points and join-irreducible elements of ${}^{\mathcal{L}}\text{supp}(f)$, i.e. $\iota(\text{supp}(f)) = \text{supp}(f) = {}^{\vee}\text{supp}(f) = {}^{\wedge}\text{supp}(f)$, if $\emptyset \notin \text{supp}(f)$ (Otherwise, we have $\iota(\text{supp}(f)) = \text{supp}(f)\backslash\{\emptyset\}$). In fact, since our iota elements are not mapped elements of a reduced set Ω' but instead raw sets from 2^{Ω}, combinations of joins lead to a vastly different lattice. In this example, we have ${}^{\mathcal{L}}\text{supp}(f) = \text{supp}(f)$, instead of the $2^{\text{supp}(f)}$ given by coarsening.

5 Conclusion

In this paper, we proposed the *Efficient Möbius Transformations* (EMT), which are general procedures to compute the zeta and Möbius transforms of any function defined on any finite distributive lattice with optimal complexity. They are based on our reformulation of the Möbius inversion theorem with focal points only, featured in our previous work [14]. The EMT optimally exploit the information contained in both the support of this function and the structure of distributive lattices. Doing so, the EMT always perform better than the optimal complexity for an algorithm considering the whole lattice, such as the FMT. Following these findings, it remains to propose explicit algorithms and implementation guidelines. We provide this for the powerset lattice, for DST, in Appendix B. We plan to release both a C++ (roughly presented in Appendix) and a Python open-source DST implementation in the near future.

Appendices

A Proofs about the Efficient Möbius Transformations

A.1 Proposition 3.1.1

Proof. Let us define L_S as the set containing the join of all combinations of iota elements of S, in addition to $\bigwedge$. Formally, we have:

$$L_S = \left\{ \bigvee X \ / \ X \subseteq \iota(S),\ X \neq \emptyset \right\} \cup \left\{ \bigwedge S \right\}.$$

By construction, we already know that the join of any number (except 0) of elements from L_S is also in L_S. So, L_S is an upper-subsemilattice of P. In the following, we will prove that it is also a lower-subsemilattice of P.

But, before anything, notice that $\bigwedge \cdot^{\uparrow S} : P \to P$ is a closure operator, i.e. for any elements $x, y \in P$, we have:

$$x \leq \bigwedge x^{\uparrow S} \tag{5}$$

$$x \leq y \ \Rightarrow \ \bigwedge x^{\uparrow S} \leq \bigwedge y^{\uparrow S} \tag{6}$$

$$\bigwedge \left(\bigwedge x^{\uparrow S} \right)^{\uparrow S} = \bigwedge x^{\uparrow S} \tag{7}$$

Now, consider the meet of two iota elements $\iota_1 \wedge \iota_2$, where $\iota_1, \iota_2 \in \iota(S)$. By definition, there are two join-irreducible elements $x, y \in {}^{\vee}\mathcal{I}(P)$ such that $\iota_1 \wedge \iota_2 = \bigwedge x^{\uparrow S} \wedge \bigwedge y^{\uparrow S} = \bigwedge (x^{\uparrow S} \cup y^{\uparrow S})$. Let us note $\delta = \bigwedge (x^{\uparrow S} \cup y^{\uparrow S})$. Since $S \supseteq x^{\uparrow S} \cup y^{\uparrow S}$, we know that $\bigwedge S \leq \delta$. If $\delta = \bigwedge S$, then $\delta \in L_S$. Otherwise, given that P is a lattice, we know that δ is equal to the join of all the join-irreducible elements of P that are less than δ, i.e. $\bigvee \delta^{\downarrow^{\vee}\mathcal{I}(P)} = \delta$.

For all join-irreducible elements $i \in \delta^{\downarrow^{\vee}\mathcal{I}(P)}$, we have $i \leq \delta$. By Eq. (6), we also have $\bigwedge i^{\uparrow S} \leq \bigwedge \delta^{\uparrow S}$. Moreover, we know that $\delta^{\uparrow S} \supseteq x^{\uparrow S} \cup y^{\uparrow S}$ because $\delta = \bigwedge (x^{\uparrow S} \cup y^{\uparrow S})$. This implies that we have $\bigwedge \delta^{\uparrow S} \leq \delta$, and so $\bigwedge i^{\uparrow S} \leq \delta$.

In addition, by Eq. (5), we get that $i \leq \bigwedge i^{\uparrow S}$, which means that, for all join-irreducible elements $i \in \delta^{\downarrow^{\vee}\mathcal{I}(P)}$, we have $i \leq \bigwedge i^{\uparrow S} \leq \delta$. Therefore, combined with the fact that $\bigvee \delta^{\downarrow^{\vee}\mathcal{I}(P)} = \delta$, we finally obtain that $\bigvee \delta^{\downarrow\iota(S)} = \delta$. In plain English, this means that δ is equal to the join of all the iota elements of S that are less than δ. So, by definition of L_S, we get that $\delta \in L_S$. Thus, the meet of two iota elements $\iota_1 \wedge \iota_2$, where $\iota_1, \iota_2 \in \iota(S)$, is in L_S.

It only remains to consider the meet of two arbitrary elements of L_S, i.e. $x \wedge y$, where $x, y \in L_S$. Notice that if $x = \bigwedge S$ or $y = \bigwedge S$, then $x \wedge y = \bigwedge S \in L_S$. Otherwise, it can be decomposed as follows:

$$x \wedge y = \left(\bigvee x^{\downarrow\iota(S)} \right) \wedge \left(\bigvee y^{\downarrow\iota(S)} \right)$$

Since P follows the distributive law Eq. (3), we can rewrite this equation as:

$$x \wedge y = \bigvee_{\iota_1 \in x^{\downarrow\iota(S)}} \ \bigvee_{\iota_2 \in y^{\downarrow\iota(S)}} (\iota_1 \wedge \iota_2)$$

The meet of two iota elements of S being in L_S, we get that $x \wedge y$ is equal to the join of elements that are all in L_S. As we already established that L_S is an upper-subsemilattice of P, we get that the meet of two arbitrary elements of L_S is also in L_S. Thus, L_S is a lower-subsemilattice of P as well and therefore a sublattice of P.

In addition, notice that for all element $s \in S$ and for all $i \in s^{\downarrow^\vee \mathcal{I}(P)}$, we have by construction $i \leq \bigwedge i^{\uparrow S} \leq s$. Therefore, P being a lattice, we have $s = \bigvee s^{\downarrow^\vee \mathcal{I}(P)} = \bigvee s^{\downarrow \iota(S)}$, i.e. $s \in L_S$. Besides, if $\bigwedge P \in S$, then it is equal to $\bigwedge S$, which is also in L_S by construction. So, $S \subseteq L_S$. It follows that the meet or join of every nonempty subset of S is in L_S, i.e. $M_S \subseteq L_S$ and $J_S \subseteq L_S$, where M_S is the smallest meet-closed subset of P containing S and J_S is the smallest join-closed subset of P containing S. Furthermore, iota elements are defined as the meet of a set of elements of S, which implies that they are necessarily all contained in M_S, i.e. $\iota(S) \subseteq M_S$. This means that we cannot build a smaller sublattice of P containing S. Therefore, L_S is the smallest sublattice of P containing S.

Finally, let us verify that all iota elements are join-irreducible elements of L_S. For any join-irreducible element $i \in {}^\vee\mathcal{I}(P)$, assume there are two distinct elements $x, y \in L_S$ such that $x < \bigwedge i^{\uparrow S}$ and $y < \bigwedge i^{\uparrow S}$. This implies by Eq. (6) and (7) that $\bigwedge x^{\uparrow S} \leq \bigwedge i^{\uparrow S}$ and $\bigwedge y^{\uparrow S} \leq \bigwedge i^{\uparrow S}$. Moreover, if $i \leq x$ and $i \leq y$, then by Eq. (6), we get that $\bigwedge i^{\uparrow S} \leq \bigwedge x^{\uparrow S}$ and $\bigwedge i^{\uparrow S} \leq \bigwedge y^{\uparrow S}$, which means that $x = y = \bigwedge i^{\uparrow S}$. This contradicts the fact that $x \neq y$. Thus, we get that $i \not\leq x$ and $i \not\leq y$. By Lemma 3.1.1, this implies that $i \not\leq x \vee y$. Since $i \leq \bigwedge i^{\uparrow S}$, we have necessarily $x \vee y \neq \bigwedge i^{\uparrow S}$ and so $x \vee y < \bigwedge i^{\uparrow S}$. Therefore, $\bigwedge i^{\uparrow S}$ is a join-irreducible element of L_S, i.e. $\iota(S)$ is the set containing only the join-irreducible elements of L_S. ■

A.2 Theorem 3.2.1

Proof. Consider the sequence $(H_k)_{k \in [\![1,n]\!]}$, where $H_k = (L, E_k)$ and:

$$E_k = \left\{(x, y) \in L^2 \ / \ y = x \vee i_{n+1-k}\right\}.$$

By definition, for all $k \in [\![1, n]\!]$ and $\forall (x, y) \in E_k$, we have $x, y \in L$ and $x \leq y$, i.e. $(x, y) \in E_{\leq}$. Reciprocally, every arrow $e \in E_{\leq}$ can be decomposed as a unique path $(e_1, e_2, \ldots, e_n) \in A_1 \times A_2 \times \cdots \times A_n$, where $A_k = E_k \cup I_P$:

Similarly to the FMT, this sequence builds unique paths simply by generating the whole lattice L step by step with each join-irreducible element of L. However, unlike the FMT, the join-irreducible elements of L are not necessarily atoms. Doing so, pairs of join-irreducible elements may be ordered, causing the sequence to skip or double some elements. And even if all the join-irreducible elements of L are atoms, since L is not necessarily a Boolean lattice, the join of two atoms may be greater than a third atom (e.g. if L is the diamond lattice), leading to the same issue. Indeed, it is easy to build a path between two elements x, y of L such that $x \leq y$: At step 1, we take the arrow $(x, x \vee i_n)$ if $i_n \leq y$ (we take the identity arrow (x, x) otherwise). At step 2, we take the arrow $(p, p \vee i_{n-1})$ if $i_{n-1} \leq y$, where $p = x \vee i_n$ or $p = x$ depending on whether or not $i_n \leq y$, and so on until we get to y. Obviously, we cannot get to y if we take an arrow $(p, p \vee i)$ where $i \not\leq y$. So, any path from x to y only consists of arrows obtained with join-irreducible elements that are less than y. But, are they all necessary to reach y from x? Let i_k be a join-irreducible element such that $i_k \leq y$. This join-irreducible element is only considered in E_{n+1-k}. Suppose we do not take the arrow at step $n + 1 - k$, i.e. we replace it by an identity arrow (p, p), where p was reached through a path from x with arrows at steps 1 to $n - k$. Since L is a distributive lattice, and since its join-irreducible elements are ordered such that $\forall i_j, i_l \in {}^\vee\mathcal{I}(L)$, $j < l \Rightarrow i_j \not\geq i_l$, we have by Corollary 3.1.1 that for any $k \in [\![1, n]\!]$, $i_k \not\leq \bigvee {}^\vee\mathcal{I}(L)_{k-1}$. So, if $i_k \not\leq p$, then by Lemma 3.1.1, we also have $i_k \not\leq p \vee \bigvee {}^\vee\mathcal{I}(L)_{k-1}$. Since

${}^{\vee}\mathcal{I}(L)_{k-1}$ contains all join-irreducible elements considered after step $n+1-k$, this implies that there is no path from p to an element greater than i_k. Yet, $i_k \leq y$. Thus, if $i_k \not\leq p$, then y can only be reached from p through a path containing the arrow $(p, p \vee i_k)$. Otherwise, if $i_k \leq p$, then $p = p \vee i_k$, which means that only an identity arrow can be taken at step $n+1-k$ anyway. Either way, there is only one arrow that can be taken at step $n+1-k$ to build a path between p and y. All join-irreducible elements less than y must be used to build a path from x to y. Thereby, this path is unique, meaning that Theorem 2.3.1 is satisfied. The sequence $(H_k)_{k\in[\![1,n]\!]}$ computes the same zeta transformations as $G_<$. ■

A.3 Theorem 3.2.2

Proof. Consider the sequence $(H_k)_{k\in[\![1,n]\!]}$, where $H_k = (M, E_k)$ and:

$$E_k = \left\{(x,y) \in M^2 \ / \ x = \bigwedge (y \vee i_k)^{\uparrow M} \text{ and } x \leq y \vee \bigvee \iota(M)_k \right\},$$

where $\iota(M)_k = \{i_1, i_2, \ldots, i_k\}$.

By definition, for all $k \in [\![1,n]\!]$ and $\forall (x,y) \in E_k$, we have $x, y \in M$ and $x \geq y$, i.e. $(x,y) \in E_{\geq}$. Reciprocally, every arrow $e \in E_{\geq}$ can be decomposed as a unique path $(e_1, e_2, \ldots, e_n) \in A_1 \times A_2 \times \cdots \times A_n$, where $A_k = E_k \cup I_P$:

Recall the procedure, described in Theorem 3.2.1, that builds unique paths simply by generating all elements of a finite distributive lattice $L \supseteq M$, based on the join of its join-irreducible elements, step by step. Here, the idea is to do the same, except that we remove all elements that are not in M. Doing so, the only difference is that the join $y \vee i_k$ of an element $y \in M$ with a join-irreducible element $i_k \in \iota(M)$ of this hypothetical lattice L may not be in M. However, thanks to the meet-closure of M, we can "jump the gap" between two elements y and p of M, should they be separated by elements of $L \backslash M$. Indeed, for all join-irreducible element $i_k \in \iota(M)$, if $x \geq y \vee i_k$, then since M is meet-closed and $x \in M$, there is a unique element $p \in M$ that we call *proxy* such that $p = \bigwedge (y \vee i_k)^{\uparrow M}$. In complement, the synchronizing condition $p \leq y \vee \bigvee \iota(M)_k$ ensures the unicity of this jump, and so the unicity of the path between any two elements x and y of M.

Finding a path from x to y is easy: take all iota elements less than x, and simply compute successive joins with them, starting from y. At step n, we take the arrow $(\bigwedge (y \vee i_n)^{\uparrow M}, y)$ if $i_n \leq x$, and (y, y) otherwise. At step $n-1$, we take the arrow $(\bigwedge (p \vee i_{n-1})^{\uparrow M}, p)$ if $i_{n-1} \leq x$, where $p = \bigwedge (y \vee i_n)^{\uparrow M}$ or $p = y$, depending on whether or not $i_n \leq x$. Proceeding as such until x is reached guarantees the existence of a path, since the synchronizing condition is always satisfied. Then, the question is: Are there other paths?

Obviously, for some $p \in M$ such that $x \geq p$, no path from x to y can contain arrows $(\bigwedge (p \vee i)^{\uparrow M}, p)$ if $i \not\leq x$. Thus, it contains only arrows corresponding to joins with iota elements that are less than x. Next, let us consider some element $p \in M$. If $i_{k-1} \leq p$, then $p = \bigwedge (p \vee i_{k-1})^{\uparrow M}$, which means that only an identity arrow (p,p) can be taken at step $k-1$. Otherwise, if $i_{k-1} \not\leq p$, then the arrow $(\bigwedge (p \vee i_{k-1})^{\uparrow M}, p)$ can only exist in the sequence if $\bigwedge (p \vee i_{k-1})^{\uparrow M} \leq p \vee \bigvee \iota(M)_{k-1}$. We know by Corollary 3.1.1 that $i_k \not\leq \bigvee \iota(M)_{k-1}$. By Lemma 3.1.1, this implies that if $i_k \not\leq p$, then $i_k \not\leq p \vee \bigvee \iota(M)_{k-1}$, which means that $i_k \not\leq \bigwedge (p \vee i_{k-1})^{\uparrow M}$. Through the same reasoning, this means that $i_k \not\leq \bigwedge \left(\bigwedge (p \vee i_{k-1})^{\uparrow M} \vee i_{k-2}\right)^{\uparrow M}$. In fact, by recurrence, we have $i_k \not\leq \bigwedge \left(\cdots \bigwedge \left(\bigwedge (p \vee i_{k-1})^{\uparrow M} \vee i_{k-2}\right)^{\uparrow M} \cdots \vee i_1\right)^{\uparrow M}$. Thus, if $i_k \not\leq p$, then no element greater than i_k can be at the tail of a path leading to p through arrows of the sequence. This means that on a path from x to y, if $i_k \leq x$, then at step k, either $i_k \leq p$

(i.e. we can only take the identity arrow (p, p)) or $i_k \not\leq p$, which implies that we must take the arrow $(\bigwedge(p \vee i_k)^{\uparrow M}, p)$. This implies that all iota elements less than x must be used in the joins corresponding to the arrows of the path from x to y. Therefore, the path described above is unique, which satisfies Theorem 2.3.1. The sequence $(H_k)_{k\in[\![1,n]\!]}$ computes the same zeta transformations as $G_>$. ■

B Implementation of the Efficient Möbius Transformations (EMT)

We present in this section evidence-based algorithms for the computation of DST transformations such as the commonality function q and the implicability function b, as well as their inversions, namely the mass function m and the conjunctive and disjunctive weight functions w and v. All these algorithms have better complexities than the FMT, i.e. less than $O(|\Omega|.2^{|\Omega|})$.

We implemented these algorithms as a general-purpose C++ *evidence-based* framework along with combination rules from DST. We plan to transpose this implementation as a Python package in the near future to ease its usage. Code and implementation details can be found at [18].

B.1 Data structure

B.1.1 Overview

The core of our implementation uses the class *bitset* (a contiguous sequence of bits) from the standard C++ library, along with a special dynamic binary tree of our design presented in section B.1.3. This tree allows us to search for supersets or subsets without having to consider all sets. For simple look-ups, i.e. just to get the value associated with a set, we use Hashmaps, since they feature constant time complexities for look-up and insertion.

Each representation of evidence (mass function, commonality function, implicability function, conjunctive weight function, disjunctive weight functions, etc) has its own class. A mass function is an object containing a tree of values different from 0, i.e. corresponding to focal sets. It inherits the abstract class *mobius_transform* which simply defines the behavior of storing values such as focal sets. An object inheriting this abstract class can be created either directly, by providing a key-value object such as an Hashmap or a tree, or indirectly, through inversion of a given zeta transform and a given order relation ($\subseteq$ or $\supseteq$). It also features methods to remove negligeable values and renormalize. The conjunctive and disjunctive weight functions also inherit this abstract class. For these last classes, all set that is not present in their tree is associated with 1.

Other representations inherit the abstract class *zeta_transform*. This abstract class defines the behavior of computing focal points from focal sets, given some order relation, and storing their values. It also keeps its mobius_transform object in memory for eventual additional projections, i.e. to get the value associated with other sets than focal points. An object inheriting this abstract class can be created either directly, by providing another zeta_transform object, or indirectly, by providing an object inheriting the mobius_transform abstract class, an order relation ($\subseteq$ or $\supseteq$) and an operation ($+$ or $\times$). For the latter, you may also provide a specific computation scheme (without structure, as a semilattice or as a lattice). This class also features methods to find the value associated with a non-focal point. The commonality, implicability

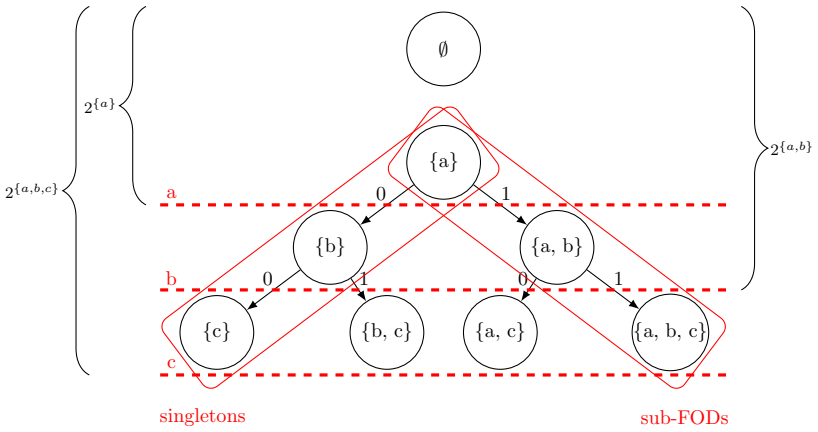

Figure 11: Illustration depicting our data structure for powerset functions on the frame of discernment $\Omega = \{a, b, c\}$. It is a binary tree with values on nodes and leaves. For the sake of clarity, the structure is shown as if it was static, where all elements from 2^Ω are considered special elements. We see clearly that it reproduces the natural generation of the powerset lattice, encapsulating the powerset of every $\{\omega_1, \ldots, \omega_n\}$ in the powerset of $\{\omega_1, \ldots, \omega_{n+1}\}$. As a consequence, the search for all singletons and all these sub-FODs can be restrained respectiveley to the left and right chain of nodes.

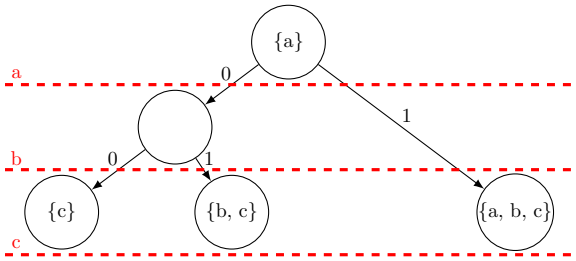

Figure 12: Illustration depicting our data structure for powerset functions on the frame of discernment $\Omega = \{a, b, c\}$. Here, special sets are $\{a\}$, $\{c\}$, $\{b, c\}$ and $\{a, b, c\}$. The blank node is a disjunction node.

and plausibility functions inherit this abstract class.

B.1.2 Frame of discernment

A frame of discernment (FOD), noted Ω, is represented as an object containing an array of labels and a Hashmap that enables one to find the index associated with a particular label.

B.1.3 Powerset function

Our data structure for *powerset functions* (i.e. functions that assign values to elements of 2^Ω) is based on the representation of sets as binary strings, as in [9] and [19], and on the binary tree depicted in Fig. 11. It is a dynamic powerset binary tree, only storing nodes corresponding to *special sets*. All other set not present in the tree is assumed to be associated with a fixed value (i.e. 0 for a mass function, 1 for a weight function). Thus, special sets include focal sets but may not all be focal sets, as is the case with the lattice support. Each node in the tree contains a boolean value to indicate whether it has been set to a value or not, an eventual *value*, pointers to parent and children, its depth index and a bitset representing an element from 2^Ω.

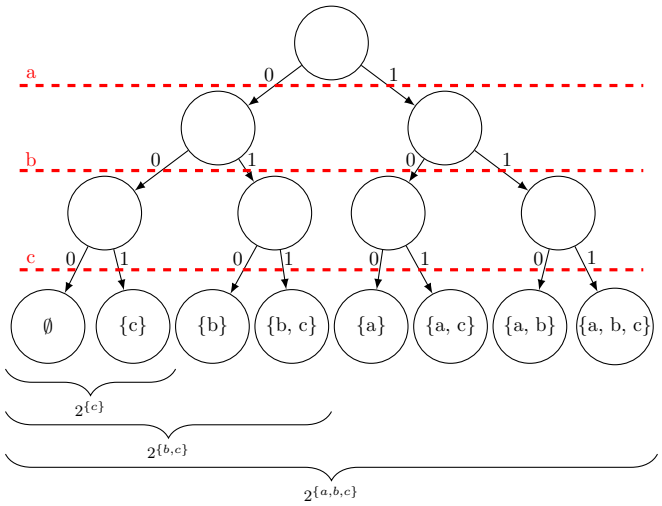

Figure 13: Same example as in Fig. 11 with an analogous data structure proposed by Wilson [9]. It is a binary tree in which values are only assigned to terminal leaves, not intermediate nodes. As in Fig. 11, for the sake of clarity, the structure is shown as if it was static, where all elements from 2^{Ω} are considered special elements.

The creation of this tree is incremental, starting with the singleton $\{\omega_1\}$, where $\Omega = [\omega_1, \ldots, \omega_n]$, as root, whether it is a special set or not. The next special set is inserted to its right or left given that it contains ω_1 or not. In fact, for a pair (A, value), where A is a special set, this procedure will assign *value* to a node of depth equal to the greatest index in $[\![1, n]\!]$ corresponding to an element of A. This node is found following the binary code that represents A to navigate the tree until the last element index is encountered in A, i.e. until we reach the last bit set to 1 in A. So, for some other special set B, if B contains all elements of A, in addition to other ones including one of greater maximum index, then B is inserted at the right of A. If B has all elements of A, in addition to other ones not including one greater than the maximum index of A, then B is not on the same branch as A and will be assigned to a node of same depth as A. All supersets of A have a depth equal to or greater than its. Of course, this also means that all subsets of A have a depth equal to or less than its. Moreover, sets that are at the left of A are not contained in A and do not contain A, since they have all elements of A but the one of greatest index, in addition to others of greater index than the maximum index of A. Furthermore, if B diverges with A before its depth, it may be necessary to create a *disjunction*. A disjunction node (i.e. a node that does not hold any value) is inserted to split the branch in two at the depth equal to the first element index not common to both A and B. The one that does not contain it will be placed at the left of this disjunction node, and the one that does will be placed at its right. This behavior is illustrated in Fig. 12.

A similar binary tree for powerset functions has been proposed in [9]. It is illustrated in Fig. 13. While both our binary tree and theirs are dynamic, theirs does not exploit depths and requires to store $2F - 1$ elements, where F is the number of special elements to store. Ours needs to store at most $F + \frac{F}{2}$ elements, and only F if every disjunction node between two special sets is also a special set (or if there is simply no disjunction between special sets, e.g. in the consonant case). Furthermore, it features interesting properties like the fact that the search for all singletons and some sub-FODs can be restrained respectiveley to the left and right chains of nodes. As mentioned above, having depths allows us to exploit the fact that subsets can only be of lower depth, while supersets can only be of greater depth. This form of powerset binary tree also speeds up the search for any value since we have to check at most n booleans to find a value, where $n \in [1, |\Omega|]$, while the version of [9] always have to check n booleans.

Algorithm 1: Computation of the focal points associated with $(S, \leq)$, where $\leq \in \{\subseteq, \supseteq\}$.

Input: S, $\leq$
Output: F, F_{Map}
$F \leftarrow S$;
$F_{\text{Map}} \leftarrow$ Hashmap<bitset, float>;
for $i = 1$ **to** $|S|$ **do**
  $F_{\text{Map}}[S[i]] \leftarrow 0$;
if $\leq \; = \; \subseteq$ **then**
  `// | is the bitwise OR operator`
  $\cdot \leftarrow |$;
else
  `// & is the bitwise AND operator`
  $\cdot \leftarrow \&$;
for $i = 1$ **to** $|S|$ **do**
  for $ii = i + 1$ **to** $|F|$ **do**
    $A \leftarrow F[ii] \cdot S[i]$;
    if $A \notin F_{Map}.keys()$ **then**
      append A to F;
      $F_{\text{Map}}[A] \leftarrow 0$;

Recently, another similar structure to ours has been proposed in [20, 21]. However, their data structure is not dynamic, i.e. it stores all subsets instead of only special sets. Doing so, they do not have to store a binary string in each node, but they always have an overall exponential spatial complexity (and of course, a time complexity at least exponential accordingly).

In the case of an infinite FOD, the idea is to represent it as an ever changing FOD containing a special element ω_∞ as first element that symbolizes the *rest* of the FOD. As we know that ω_∞ will always be an element of this FOD, having $\{\omega_\infty\}$ as the root node of its powerset binary tree reduces the number of operations of reorganization after addition or removal of any FOD element.

B.2 Procedures computing focal points

In this section, we present algorithms computing all focal points, given focal sets.

B.2.1 General procedure

Here, we apply Property 2 of section 3.4 of [14] to the case where $P = 2^\Omega$, leading to Algorithm 1. For a set $S \subseteq 2^\Omega$, e.g. $\text{supp}(f)$, this algorithm computes its focal points ${}^oS$, where $o \in \{\wedge, \vee\}$ with a complexity less than $O(|S| \; . \; |{}^oS|)$ in time and $O(|{}^oS|)$ in space.

B.2.2 Linear analysis

There are some cases in which a linear pre-analysis of complexity $O(|S|)$ both in time and space is sufficient to find all focal points. This analysis focuses on the progressive union of all focal sets in a linear run. Formally, let $S\backslash\{\Omega\} = \{A_1, A_2, \ldots, A_K\}$ and $S\backslash\{\Omega\}_k = \{A_1, A_2, \ldots, A_k\}$. This analysis focuses on $I_i = U_{i-1} \cap A_i$ such that $i \in [\![2, K]\!]$, and $U_{i-1} = \bigcup S\backslash\{\Omega\}_{i-1}$ and $I_1 = A_1$. More precisely, we have

$$\begin{aligned} I_i &= U_{i-1} \cap A_i \\ &= (\textstyle\bigcup S\backslash\{\Omega\}_{i-1}) \ \cap A_i \\ &= \textstyle\bigcup_{F \in \{A_1, \ldots, A_{i-1}\}} (F \cap A_i)\,. \end{aligned}$$

In other words, these intersections I_i contain all focal points based on pairs of focal sets.

In most cases, without testing each pair of focal sets, we cannot know which focal points are generated based solely on I_i since several combinations of sets in 2^{I_i} can lead to I_i as union. For example, if $\Omega = \{a, b, c, d\}$ and $S\backslash\{\Omega\} = \{\{a, b\}, \{a, c\}, \{b, c, d\}\}$, then $U_2 = \{a, b, c\}$ and $I_3 = \{a, b, c\} \cap \{b, c, d\} = \{b, c\}$. Yet, $\{b, c\}$ is not a focal point. It is the result of the union of the focal points $\{a, b\} \cap \{b, c, d\} = \{b\}$ and $\{a, c\} \cap \{b, c, d\} = \{c\}$.

However, there are two special cases in which we know that I_i is a focal point: (a) if $|I_i| = 0$, then $2^{I_i} = \{I_i\}$, and (b) if $|I_i| = 1$, then $2^{I_i} = \{I_i, \emptyset\}$.

So, if $\forall i \in [\![2, K]\!]$, $|I_i| = 0$, then $\forall F_1, F_2 \in S\backslash\{\Omega\}$, $F_1 \cap F_2 = \emptyset$. This is the *quasi-Bayesian* case that has already been treated in Proposition 1 of [22]. By definition of this case, we have $\forall F \in S\backslash\{\emptyset, \Omega\}$, $F^{\uparrow\overset{\wedge}{}S} = \{\Omega\}$ and the only possible focal point other than the focal sets is $\emptyset$, i.e. $^{\wedge}S\backslash\{\emptyset\} = S\backslash\{\emptyset\}$. This means that for any powerset function f such that $\mathrm{supp}(f) \subseteq S$, the computation of its zeta and Möbius transforms in $(2^{\Omega}, \supseteq)$ is always $O(|\mathrm{supp}(f)|)$, where $|\mathrm{supp}(f)| \leq |\Omega| + 2$.

In addition, this analysis points to a slightly more general case that contains the quasi-Bayesian one in which $\forall i \in [\![2, K]\!]$, $|I_i| \leq 1$. Indeed, when $|I_i| = 1$, we have $2^{I_i} = \{I_i, \emptyset\}$, which means that all focal points composing I_i are in $\{I_i, \emptyset\}$ and at least one of them is the singleton I_i, since $\emptyset$ is the neutral element for the union of sets. Also, the only new focal point that could be generated based on the intersection of the singleton I_i with another focal point is $\emptyset$.

Algorithm 2 sums up this procedure for $^{\wedge}S$. This linear analysis can be performed for the dual order $\subseteq$ as well, focusing on $S\backslash\{\emptyset\} = \{A_1, A_2, \ldots, A_K\}$, with $\overline{I_i} = \overline{U_{i-1}} \cup A_i$ such that $i \in [\![2, K]\!]$, and $\overline{U_{i-1}} = \bigcap S\backslash\{\emptyset\}_{i-1}$ and $\overline{I_1} = A_1$. This dual procedure is provided by Algorithm 3.

B.3 Computation of iota elements

Computation of the iota elements (See Proposition 3.1.1) and dual iota elements (See Proposition 3.1.2) of S are presented respectively in Algorithms 4 and 5.

B.4 Computation of the lattice support

Computation of the upper and lower closure in the lattice support of S (See Proposition 3.1.3) are presented respectively in Algorithms 6 and 7.

Algorithm 2: Linear computation of $^{\wedge}S$ based on S.

Input: $S\backslash\{\Omega\} = \{A_1, A_2, \ldots, A_K\}$
Output: $^{\wedge}S$, *is_almost_bayesian*
$^{\wedge}S \leftarrow S$;
$U \leftarrow A_1$;
is_almost_bayesian $\leftarrow$ True;
for $i = 2$ **to** K **do**
    $I \leftarrow U \cap A_i$;
    if $|I| > 1$ **then**
        is_almost_bayesian $\leftarrow$ False;
        break;
    else if $|I| = 1$ **then**
        if $I \notin {}^{\wedge}S$ **then**
            append I to $^{\wedge}S$;
    if $\emptyset \notin {}^{\wedge}S$ **then**
        append $\emptyset$ to $^{\wedge}S$;
    $U \leftarrow U \cup A_i$;

Algorithm 3: Linear computation of $^{\vee}S$ based on S.

Input: $S\backslash\{\emptyset\} = \{A_1, A_2, \ldots, A_K\}$
Output: $^{\vee}S$, *is_almost_bayesian*
$^{\vee}S \leftarrow S$;
$U \leftarrow A_1$;
is_almost_bayesian $\leftarrow$ True;
for $i = 2$ **to** K **do**
    $I \leftarrow U \cup A_i$;
    if $|I| < |\Omega| - 1$ **then**
        is_almost_bayesian $\leftarrow$ False;
        break;
    else if $|I| = |\Omega| - 1$ **then**
        if $I \notin {}^{\vee}S$ **then**
            append I to $^{\vee}S$;
    if $\Omega \notin {}^{\vee}S$ **then**
        append Ω to $^{\vee}S$;
    $U \leftarrow U \cap A_i$;

Algorithm 4: Computation of $\iota(S)$.

Input: S
Output: I
foreach $\omega \in \Omega$ **do**
  $i \leftarrow \Omega$;
  foreach $F \in S$, *such that* $F \supseteq \{\omega\}$ **do**
    included $\leftarrow$ True;
    $i \leftarrow i \cap F$;
    if $i = \{\omega\}$ **then**
      break;
  if *included* **then**
    append i to I;

Algorithm 5: Computation of $\overline{\iota}(S)$.

Input: S
Output: I
foreach $\omega \in \Omega$ **do**
  $i \leftarrow \emptyset$;
  foreach $F \in S$, *such that* $F \subseteq \Omega \backslash \{\omega\}$ **do**
    included $\leftarrow$ True;
    $i \leftarrow i \cup F$;
    if $i = \Omega \backslash \{\omega\}$ **then**
      break;
  if *included* **then**
    append i to I;

Algorithm 6: Computation of $S^{\uparrow^{\mathcal{L}} S}$ based on $\iota(S)$ and S.

Input: $\iota(S)$, S
Output: L
$L \leftarrow S$;
foreach $i \in \iota(S)$ **do**
  foreach $A \in L$ **do**
    $B \leftarrow A \cup i$;
    if $B \notin L$ **then**
      append B to L;

B.5 Computation of DST transformations in the consonant case

The consonant case has already been treated in Proposition 2 of [22]. A consonant structure of evidence is a nested structure where each focal set is contained in every focal set of greater or equal cardinality. Formally, $\forall F_i, F_j \in \mathcal{F}$, if $|F_i| \leq |F_j|$ then $F_i \subseteq F_j$. By definition, there can be only one focal set of each cardinality in $[\![0, |\Omega|]\!]$, each of them being a subset of every focal set of greater cardinality. This means that there can be no focal point other than focal sets and that

Algorithm 7: Computation of $S^{\downarrow^{\mathcal{L}}S}$ based on $\overline{\iota}(S)$ and S.

Input: $\overline{\iota}(S)$, S
Output: L
$L \leftarrow S$;
foreach $i \in \overline{\iota}(S)$ **do**
  foreach $A \in L$ **do**
    $B \leftarrow A \cap i$;
    if $B \notin L$ **then**
      append B to L;

each focal point has a proxy focal point. Let us note $|F_1| < |F_2| < \cdots < |F_K|$ the focal elements in S. To check if this structure is consonant, we simply have to check that $\forall i \in [\![1, K-1]\!]$, $F_i \subset F_{i+1}$. This analysis is performed by Algorithm 8.

Algorithms 9 to 16 present procedures in the consonant case to compute the following transformations:

- m to b,
- m to q,
- b to v,
- q to w,
- b to m,
- q to m,
- v to b,
- w to q.

Their complexity in time is $\min\left[O(|\Omega|), O\left(|S|.\log\left(|S|\right)\right)\right]$ (which corresponds to the complexity of the sorting algorithm used on S). Their space complexity is $O(|S|)$.

Algorithm 8: Consonance check.

Input: S
Output: *is_consonant*
sort S such that $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$;
is_consonant $\leftarrow$ True;
for $i \in [\![1, K-1]\!]$ **do**
  if $F_i \not\subseteq F_{i+1}$ **then**
    is_consonant $\leftarrow$ False;
    break;

B.6 Computation of DST transformations in a semilattice

Algorithms 17 to 24 present procedures using ${}^{\wedge}S$ or ${}^{\vee}S$ to compute the following transformations:

- m to b,
- m to q,
- b to v,
- q to w,
- b to m,
- q to m,
- v to b,
- w to q.

They all use the sequences of graphs of Theorem 3.2.2 and Corollary 3.2.7. Their complexity in time is $O(I(S).|^{o}S|.\epsilon)$, where $I \in \{\iota, \overline{\iota}\}$ and $o \in \{\wedge, \vee\}$ and where ϵ represents the average number of operations required to "bridge a gap" (See Theorem 3.2.2). This is always less than $O(|\Omega|.2^{\Omega})$. Their space complexity is $O(|^{o}S|)$.

B.7 Computation of DST transformations in a lattice

The general procedure to compute ${}^{o}S$ is less than $O(|S|.|{}^{o}S|)$. But, if one wishes to be certain that our algorithms are at least as efficient as the FMT, i.e. at most $O(|\Omega|.2^{|\Omega|})$, then one has to look at $|S|$. Indeed, if $O(|S|) \leq O(|\Omega|)$ (e.g. $|S| < 10.|\Omega|$), then $O(|S|.|{}^{o}S|) \leq O(|\Omega|.2^{|\Omega|})$. Otherwise, one can rely on the lattice support ${}^{\mathcal{L}}S$, since $\iota(S)$ can be computed in less than $O(|\Omega|.|S|)$ with Algorithm 4 and is then used to compute ${}^{o}S$ in $O(\iota(S).|{}^{\mathcal{L}}S|)$ in Algorithms 6 and 7, where $|\iota(S)| \leq |\Omega|$.

Algorithms 25 to 32 present procedures using ${}^{\mathcal{L}}S$ (more precisely, the upper and lower closures $S^{\uparrow^{\mathcal{L}}S}$ and $S^{\downarrow^{\mathcal{L}}S}$) to compute the following transformations:

- m to b,
- m to q,
- b to v,
- q to w,
- b to m,
- q to m,
- v to b,
- w to q.

They all use the sequences of graphs of Theorem 3.2.1 and its corollaries. Their complexity in time is less than $O(I(S).|{}^{\mathcal{L}}S|)$, where $I \in \{\iota, \bar{\iota}\}$ and $o \in \{\wedge, \vee\}$, which is always less than $O(|\Omega|.2^{|\Omega|})$. Their space complexity is less than $O(|{}^{\mathcal{L}}S|)$.

B.8 Computation of DST transformations independently from Ω

If $\text{supp}(f)$ is almost Bayesian or if $|\Omega|$ happens to be considerable to the point that one would like to compute the previous DST transformations independently from $|\Omega|$, Algorithms 33 to 40 present procedures of time complexities in $[O(|{}^{o}S|), O(|{}^{o}S|^2)]$. Their complexity in space is $O(|{}^{o}S|)$. They present procedures for the following transformations:

- m to b,
- m to q,
- b to v,
- q to w,
- b to m,
- q to m,
- v to b,
- w to q.

Fig. 14 offers a decision tree to help the reader in choosing the right algorithm for their case. Of course, this decision tree can be implemented as an algorithm in order to create a unique general procedure automatically choosing the type of algorithm to use, no matter the DST transformation.

Algorithm 9: Computation of $\{b(B) \ / \ B \in S \ \}$ based on $\{m(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{m(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$
Output: $\{b(B) \ / \ B \in S \ \}$
if *is_consonant* **then**
  $b(F_1) \leftarrow m(F_1)$;
  for $i = 2$ **to** K **do**
    $b(F_i) \leftarrow m(F_i) + b(F_{i-1})$;

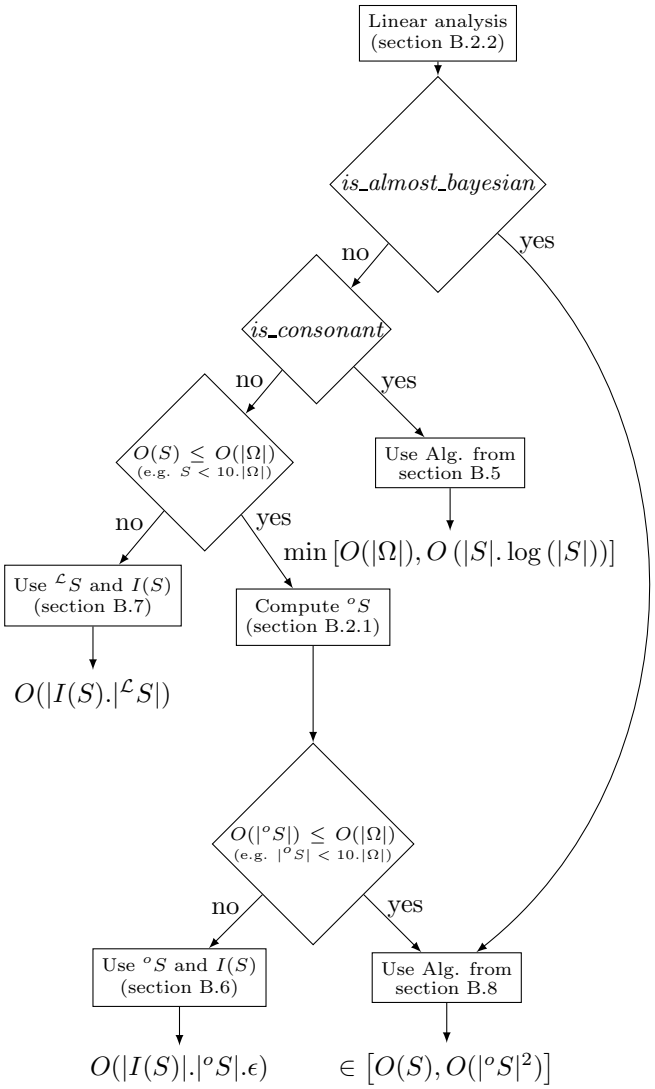

Figure 14: Decision tree for the choice of which algorithms of section B to use. Of course, this decision tree can be implemented as an algorithm in order to create a unique general procedure automatically choosing the type of algorithm to use, no matter the DST transformation. Diamond nodes represent Boolean tests. Rectangle nodes represent the chosen action. Finally, terminal nodes (leaves) indicate the final time complexity of the whole procedure, including the computation of any DST transformation, where $o \in \{\wedge, \vee\}$ and $I \in \{\iota, \overline{\iota}\}$. In particular, when *is_almost_Bayesian* is true, the complexity of the whole procedure is $O(S)$. Notice that all these complexities are less than $O(|\Omega|.2^{|\Omega|})$.

Algorithm 10: Computation of $\{m(B) \ / \ B \in S \ \}$ based on $\{b(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{b(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{m(B) \ / \ B \in S \ \}$

if *is_consonant* **then**
- $m(F_1) \leftarrow b(F_1)$;
- **for** $i = 2$ **to** K **do**
  - $m(F_i) \leftarrow b(F_i) - b(F_{i-1})$;

Algorithm 11: Computation of $\{q(B) \ / \ B \in S \ \}$ based on $\{m(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{m(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{q(B) \ / \ B \in S \ \}$

if *is_consonant* **then**
- $q(F_K) \leftarrow m(F_K)$;
- **for** $i = K - 1$ **to** 1 **do**
  - $q(F_i) \leftarrow m(F_i) + q(F_{i+1})$;

Algorithm 12: Computation of $\{m(B) \ / \ B \in S \ \}$ based on $\{q(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{q(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{m(B) \ / \ B \in S \ \}$

if *is_consonant* **then**
- $m(F_K) \leftarrow q(F_K)$;
- **for** $i = K - 1$ **to** 1 **do**
  - $m(F_i) \leftarrow q(F_i) - q(F_{i+1})$;

Algorithm 13: Computation of $\{v(B) \ / \ B \in S \ \}$ based on $\{b(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{b(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{v(B) \ / \ B \in S \ \}$

if *is_consonant* **then**
- $v(F_1) \leftarrow b(F_1)^{-1}$;
- **for** $i = 2$ **to** K **do**
  - $v(F_i) \leftarrow b(F_i)^{-1}.\ b(F_{i-1})$;

Algorithm 14: Computation of $\{b(B) \ / \ B \in S \ \}$ based on $\{v(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{v(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{b(B) \ / \ B \in S \ \}$

if *is_consonant* **then**

  $b(F_1) \leftarrow v(F_1)^{-1}$;

  for $i = 2$ **to** K **do**

    $b(F_i) \leftarrow v(F_i)^{-1}.\ b(F_{i-1})$;

Algorithm 15: Computation of $\{w(B) \ / \ B \in S \ \}$ based on $\{q(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{q(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{w(B) \ / \ B \in S \ \}$

if *is_consonant* **then**

  $w(F_K) \leftarrow q(F_K)^{-1}$;

  for $i = K - 1$ **to** 1 **do**

    $w(F_i) \leftarrow q(F_i)^{-1}.\ q(F_{i+1})$;

Algorithm 16: Computation of $\{q(B) \ / \ B \in S \ \}$ based on $\{w(B) \ / \ B \in S \ \}$ in the consonant case.

Input: $\{w(B) \ / \ B \in S \ \}$, *is_consonant*, $S = \{F_1, F_2, \ldots, F_K\}$, where $|F_1| \leq |F_2| \leq \cdots \leq |F_K|$

Output: $\{q(B) \ / \ B \in S \ \}$

if *is_consonant* **then**

  $q(F_K) \leftarrow w(F_K)^{-1}$;

  for $i = K - 1$ **to** 1 **do**

    $q(F_i) \leftarrow w(F_i)^{-1}.\ q(F_{i+1})$;

Algorithm 17: Computation of $\{b(B) \ / \ B \in {}^{\vee}S \ \}$ based on $\{m(B) \ / \ B \in {}^{\vee}S \ \}$.

Input: $\{m(B) \ / \ B \in {}^{\vee}S \ \}$, ${}^{\vee}S$, $\overline{\iota}(S)$
Output: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^{\vee}S$ **do**
    $b(A) \leftarrow m(A)$;
$\Omega_{cum} \leftarrow \Omega$;
for $k = K$ **to** 1 **do**
    $\Omega_{cum} \leftarrow \Omega_{cum} \cap i_k$;
    foreach $A \in {}^{\vee}S$ **do**
        $B \leftarrow A \cap i_k$;
        if $B \neq A$ **then**
            $X \leftarrow \underset{C \in B^{\downarrow^{\vee}S}}{\operatorname{argmax}} (|C|)$;
            if $X \neq NULL$ ***and*** $X \supseteq A \cap \Omega_{cum}$ **then**
                $b(A) \leftarrow b(A) + b(X)$;

Algorithm 18: Computation of $\{m(B) \ / \ B \in {}^{\vee}S \ \}$ based on $\{b(B) \ / \ B \in {}^{\vee}S \ \}$.

Input: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$, ${}^{\vee}S$, $\overline{\iota}(S)$
Output: $\{m(B) \ / \ B \in {}^{\vee}S \ \}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^{\vee}S$ **do**
    $m(A) \leftarrow b(A)$;
$\Omega_{cum} \leftarrow \Omega$;
for $k = 1$ **to** K **do**
    $\Omega_{cum} \leftarrow \Omega_{cum} \cap i_k$;
    foreach $A \in {}^{\vee}S$ **do**
        $B \leftarrow A \cap i_k$;
        if $B \neq A$ **then**
            $X \leftarrow \underset{C \in B^{\downarrow^{\vee}S}}{\operatorname{argmax}} (|C|)$;
            if $X \neq NULL$ ***and*** $X \supseteq A \cap \Omega_{cum}$ **then**
                $m(A) \leftarrow m(A) - m(X)$;

Algorithm 19: Computation of $\{q(B) \ / \ B \in {}^\wedge S \ \}$ based on $\{m(B) \ / \ B \in {}^\wedge S \ \}$.

Input: $\{m(B) \ / \ B \in {}^\wedge S \ \}$, ${}^\wedge S$, $\iota(S)$
Output: $\{q(B) \ / \ B \in {}^\wedge S \ \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^\wedge S$ **do**
  $q(A) \leftarrow m(A)$;
$\Omega_{cum} \leftarrow \emptyset$;
for $k = 1$ **to** K **do**
  $\Omega_{cum} \leftarrow \Omega_{cum} \cup i_k$;
  foreach $A \in {}^\wedge S$ **do**
    $B \leftarrow A \cup i_k$;
    if $B \neq A$ **then**
      $X \leftarrow \underset{C \in B^{\uparrow \wedge S}}{\text{argmin}} \, (|C|)$;
      if $X \neq NULL$ ***and*** $X \subseteq A \cup \Omega_{cum}$ **then**
        $q(A) \leftarrow q(A) + q(X)$;

Algorithm 20: Computation of $\{m(B) \ / \ B \in {}^\wedge S \ \}$ based on $\{q(B) \ / \ B \in {}^\wedge S \ \}$.

Input: $\{q(B) \ / \ B \in {}^\wedge S \ \}$, ${}^\wedge S$, $\iota(S)$
Output: $\{m(B) \ / \ B \in {}^\wedge S \ \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^\wedge S$ **do**
  $m(A) \leftarrow q(A)$;
$\Omega_{cum} \leftarrow \emptyset$;
for $k = K$ **to** 1 **do**
  $\Omega_{cum} \leftarrow \Omega_{cum} \cup i_k$;
  foreach $A \in {}^\wedge S$ **do**
    $B \leftarrow A \cup i_k$;
    if $B \neq A$ **then**
      $X \leftarrow \underset{C \in B^{\uparrow \wedge S}}{\text{argmin}} \, (|C|)$;
      if $X \neq NULL$ ***and*** $X \subseteq A \cup \Omega_{cum}$ **then**
        $m(A) \leftarrow m(A) - m(X)$;

Algorithm 21: Computation of $\{v(B) \ / \ B \in {}^{\vee}S \ \}$ based on $\{b(B) \ / \ B \in {}^{\vee}S \ \}$.

Input: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$, ${}^{\vee}S$, $\overline{\iota}(S)$
Output: $\{v(B) \ / \ B \in {}^{\vee}S \ \}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^{\vee}S$ **do**
    $v(A) \leftarrow b(A)^{-1}$;
$\Omega_{cum} \leftarrow \Omega$;
for $k = 1$ **to** K **do**
    $\Omega_{cum} \leftarrow \Omega_{cum} \cap i_k$;
    foreach $A \in {}^{\vee}S$ **do**
        $B \leftarrow A \cap i_k$;
        if $B \neq A$ **then**
            $X \leftarrow \underset{C \in B^{\downarrow^{\vee}S}}{\operatorname{argmax}} (|C|)$;
            if $X \neq$ *NULL* ***and*** $X \supseteq A \cap \Omega_{cum}$ **then**
                $v(A) \leftarrow v(A)/v(X)$;

Algorithm 22: Computation of $\{b(B) \ / \ B \in {}^{\vee}S \ \}$ based on $\{v(B) \ / \ B \in {}^{\vee}S \ \}$.

Input: $\{v(B) \ / \ B \in {}^{\vee}S \ \}$, ${}^{\vee}S$, $\overline{\iota}(S)$
Output: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^{\vee}S$ **do**
    $b(A) \leftarrow v(A)^{-1}$;
$\Omega_{cum} \leftarrow \Omega$;
for $k = K$ **to** 1 **do**
    $\Omega_{cum} \leftarrow \Omega_{cum} \cap i_k$;
    foreach $A \in {}^{\vee}S$ **do**
        $B \leftarrow A \cap i_k$;
        if $B \neq A$ **then**
            $X \leftarrow \underset{C \in B^{\downarrow^{\vee}S}}{\operatorname{argmax}} (|C|)$;
            if $X \neq$ *NULL* ***and*** $X \supseteq A \cap \Omega_{cum}$ **then**
                $b(A) \leftarrow b(A).b(X)$;

Algorithm 23: Computation of $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$ based on $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$.

Input: $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$, ${}^{\wedge}S$, $\iota(S)$
Output: $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^{\wedge}S$ **do**
    $w(A) \leftarrow q(A)^{-1}$;
$\Omega_{cum} \leftarrow \emptyset$;
for $k = K$ **to** 1 **do**
    $\Omega_{cum} \leftarrow \Omega_{cum} \cup i_k$;
    foreach $A \in {}^{\wedge}S$ **do**
        $B \leftarrow A \cup i_k$;
        if $B \neq A$ **then**
            $X \leftarrow \underset{C \in B^{\uparrow^{\wedge}S}}{\text{argmin}} \, (|C|)$;
            if $X \neq NULL$ ***and*** $X \subseteq A \cup \Omega_{cum}$ **then**
                $w(A) \leftarrow w(A)/w(X)$;

Algorithm 24: Computation of $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$ based on $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$.

Input: $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$, ${}^{\wedge}S$, $\iota(S)$
Output: $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in {}^{\wedge}S$ **do**
    $q(A) \leftarrow w(A)^{-1}$;
$\Omega_{cum} \leftarrow \emptyset$;
for $k = 1$ **to** K **do**
    $\Omega_{cum} \leftarrow \Omega_{cum} \cup i_k$;
    foreach $A \in {}^{\wedge}S$ **do**
        $B \leftarrow A \cup i_k$;
        if $B \neq A$ **then**
            $X \leftarrow \underset{C \in B^{\uparrow^{\wedge}S}}{\text{argmin}} \, (|C|)$;
            if $X \neq NULL$ ***and*** $X \subseteq A \cup \Omega_{cum}$ **then**
                $q(A) \leftarrow q(A).q(X)$;

Algorithm 25: Computation of $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$ based on $\{m(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$.

Input: $\{m(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$, $S^{\downarrow^{\mathcal{L}}S}$, $\overline{\iota}(S)$
Output: $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\downarrow^{\mathcal{L}}S}$ **do**
  $b(A) \leftarrow m(A)$;
for $k = K$ **to** 1 **do**
  foreach $A \in S^{\downarrow^{\mathcal{L}}S}$ **do**
    $B \leftarrow A \cap i_k$;
    if $B \neq A$ **then**
      $b(A) \leftarrow b(A) + b(B)$;

Algorithm 26: Computation of $\{m(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$ based on $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$.

Input: $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$, $S^{\downarrow^{\mathcal{L}}S}$, $\overline{\iota}(S)$
Output: $\{m(B) \ / \ B \in S^{\downarrow^{\mathcal{L}}S}\}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\downarrow^{\mathcal{L}}S}$ **do**
  $m(A) \leftarrow b(A)$;
for $k = 1$ **to** K **do**
  foreach $A \in S^{\downarrow^{\mathcal{L}}S}$ **do**
    $B \leftarrow A \cap i_k$;
    if $B \neq A$ **then**
      $m(A) \leftarrow m(A) - m(B)$;

Algorithm 27: Computation of $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S}\}$ based on $\{m(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S}\}$.

Input: $\{m(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S}\}$, $S^{\uparrow^{\mathcal{L}}S}$, $\iota(S)$
Output: $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S}\}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\uparrow^{\mathcal{L}}S}$ **do**
  $q(A) \leftarrow m(A)$;
for $k = 1$ **to** K **do**
  foreach $A \in S^{\uparrow^{\mathcal{L}}S}$ **do**
    $B \leftarrow A \cup i_k$;
    if $B \neq A$ **then**
      $q(A) \leftarrow q(A) + q(B)$;

Algorithm 28: Computation of $\{m(B) \ / \ B \in S^{\uparrow^{\mathcal{L}} S} \}$ based on $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}} S} \}$.

Input: $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}} S} \}$, $S^{\uparrow^{\mathcal{L}} S}$, $\iota(S)$
Output: $\{m(B) \ / \ B \in S^{\uparrow^{\mathcal{L}} S} \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\uparrow^{\mathcal{L}} S}$ **do**
  $m(A) \leftarrow q(A)$;
for $k = K$ **to** 1 **do**
  foreach $A \in S^{\uparrow^{\mathcal{L}} S}$ **do**
    $B \leftarrow A \cup i_k$;
    if $B \neq A$ **then**
      $m(A) \leftarrow m(A) - m(B)$;

Algorithm 29: Computation of $\{v(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$ based on $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$.

Input: $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$, $S^{\downarrow^{\mathcal{L}} S}$, $\overline{\iota}(S)$
Output: $\{v(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\downarrow^{\mathcal{L}} S}$ **do**
  $v(A) \leftarrow b(A)^{-1}$;
for $k = 1$ **to** K **do**
  foreach $A \in S^{\downarrow^{\mathcal{L}} S}$ **do**
    $B \leftarrow A \cap i_k$;
    if $B \neq A$ **then**
      $v(A) \leftarrow v(A)/v(B)$;

Algorithm 30: Computation of $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$ based on $\{v(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$.

Input: $\{v(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$, $S^{\downarrow^{\mathcal{L}} S}$, $\overline{\iota}(S)$
Output: $\{b(B) \ / \ B \in S^{\downarrow^{\mathcal{L}} S} \}$
sort $\overline{\iota}(S)$ such that $\overline{\iota}(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\downarrow^{\mathcal{L}} S}$ **do**
  $b(A) \leftarrow v(A)^{-1}$;
for $k = K$ **to** 1 **do**
  foreach $A \in S^{\downarrow^{\mathcal{L}} S}$ **do**
    $B \leftarrow A \cap i_k$;
    if $B \neq A$ **then**
      $b(A) \leftarrow b(A).b(B)$;

Algorithm 31: Computation of $\{w(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$ based on $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$.

Input: $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$, $S^{\uparrow^{\mathcal{L}}S}$, $\iota(S)$
Output: $\{w(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\uparrow^{\mathcal{L}}S}$ **do**
  $w(A) \leftarrow q(A)^{-1}$;
for $k = K$ **to** 1 **do**
  foreach $A \in S^{\uparrow^{\mathcal{L}}S}$ **do**
    $B \leftarrow A \cup i_k$;
    if $B \neq A$ **then**
      $w(A) \leftarrow w(A)/w(B)$;

Algorithm 32: Computation of $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$ based on $\{w(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$.

Input: $\{w(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$, $S^{\uparrow^{\mathcal{L}}S}$, $\iota(S)$
Output: $\{q(B) \ / \ B \in S^{\uparrow^{\mathcal{L}}S} \}$
sort $\iota(S)$ such that $\iota(S) = \{i_1, i_2, \ldots, i_K\}$, where $|i_1| \leq |i_2| \leq \cdots \leq |i_K|$;
foreach $A \in S^{\uparrow^{\mathcal{L}}S}$ **do**
  $q(A) \leftarrow w(A)^{-1}$;
for $k = 1$ **to** K **do**
  foreach $A \in S^{\uparrow^{\mathcal{L}}S}$ **do**
    $B \leftarrow A \cup i_k$;
    if $B \neq A$ **then**
      $q(A) \leftarrow q(A).q(B)$;

Algorithm 33: Computation of $\{b(B) \ / \ B \in {}^{\vee}S \}$ based on $\{m(B) \ / \ B \in S \}$ independently from $|\Omega|$.

Input: $\{m(F) \ / \ F \in S \}$
Output: $\{b(B) \ / \ B \in {}^{\vee}S \}$
foreach $B \in {}^{\vee}S$ **do**
  $b(B) \leftarrow m(B)$;
  foreach $F \in S \ / \ F \subset B$ **do**
    $b(B) \leftarrow b(B) + m(F)$;

Algorithm 34: Computation of $\{m(B) \ / \ B \in {}^{\vee}S\}$ based on $\{b(B) \ / \ B \in {}^{\vee}S \ \}$ independently from $|\Omega|$.

Input: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$
Output: $\{m(B) \ / \ B \in {}^{\vee}S \ \}$
sort ${}^{\vee}S$ such that ${}^{\vee}S = \{A_1, A_2, \ldots, A_K\}$, where $|A_1| \leq |A_2| \leq \cdots \leq |A_K|$;
for $i = 1$ **to** K **do**
    $m(A_i) \leftarrow b(A_i)$;
    foreach $B \in {}^{\vee}S \ / \ B \subset A_i$ **do**
        $m(A_i) \leftarrow m(A_i) - m(B)$;

Algorithm 35: Computation of $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$ based on $\{m(B) \ / \ B \in S \ \}$ independently from $|\Omega|$.

Input: $\{m(F) \ / \ F \in S \ \}$
Output: $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$
foreach $B \in {}^{\wedge}S$ **do**
    $q(B) \leftarrow m(B)$;
    foreach $F \in S \ / \ F \supset B$ **do**
        $q(B) \leftarrow q(B) + m(F)$;

Algorithm 36: Computation of $\{m(B) \ / \ B \in {}^{\wedge}S\}$ based on $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$ independently from $|\Omega|$.

Input: $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$
Output: $\{m(B) \ / \ B \in {}^{\wedge}S \ \}$
sort ${}^{\wedge}S$ such that ${}^{\wedge}S = \{A_1, A_2, \ldots, A_K\}$, where $|A_1| \leq |A_2| \leq \cdots \leq |A_K|$;
for $i = K$ **to** 1 **do**
    $m(A_i) \leftarrow q(A_i)$;
    foreach $B \in {}^{\wedge}S \ / \ B \supset A_i$ **do**
        $m(A_i) \leftarrow m(A_i) - m(B)$;

Algorithm 37: Computation of $\{v(B) \ / \ B \in {}^{\vee}S \ \}$ based on $\{b(B) \ / \ B \in {}^{\vee}S \ \}$ independently from $|\Omega|$.

Input: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$, ${}^{\vee}S$
Output: $\{w(B) \ / \ B \in {}^{\vee}S \ \}$
sort ${}^{\vee}S$ such that ${}^{\vee}S = \{A_1, A_2, \ldots, A_K\}$, where $|A_1| \leq |A_2| \leq \cdots \leq |A_K|$;
for $i = 1$ **to** K **do**
    $v(A_i) \leftarrow b(A_i)^{-1}$;
    foreach $B \in {}^{\vee}S \ / \ B \subset A_i$ **do**
        $v(A_i) \leftarrow v(A_i)/v(B)$;

Algorithm 38: Computation of $\{b(B) \ / \ B \in {}^{\vee}S \ \}$ based on $\{v(B) \ / \ B \in {}^{\vee}S \ \}$ independently from $|\Omega|$.

Input: $\{v(B) \ / \ B \in {}^{\vee}S \ \}$, ${}^{\vee}S$
Output: $\{b(B) \ / \ B \in {}^{\vee}S \ \}$
foreach $B \in {}^{\vee}S$ **do**
    $b(B) \leftarrow v(B)^{-1}$;
    foreach $A \in {}^{\vee}S \ / \ A \subset B$ **do**
        $b(B) \leftarrow b(B)/v(A)$;

Algorithm 39: Computation of $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$ based on $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$ independently from $|\Omega|$.

Input: $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$, ${}^{\wedge}S$
Output: $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$
sort ${}^{\wedge}S$ such that ${}^{\wedge}S = \{A_1, A_2, \ldots, A_K\}$, where $|A_1| \leq |A_2| \leq \cdots \leq |A_K|$;
for $i = K$ **to** 1 **do**
    $w(A_i) \leftarrow q(A_i)^{-1}$;
    foreach $B \in {}^{\wedge}S \ / \ B \supset A_i$ **do**
        $w(A_i) \leftarrow w(A_i)/w(B)$;

Algorithm 40: Computation of $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$ based on $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$ independently from $|\Omega|$.

Input: $\{w(B) \ / \ B \in {}^{\wedge}S \ \}$, ${}^{\wedge}S$
Output: $\{q(B) \ / \ B \in {}^{\wedge}S \ \}$
foreach $B \in {}^{\wedge}S$ **do**
    $q(B) \leftarrow w(B)^{-1}$;
    foreach $A \in {}^{\wedge}S \ / \ A \supset B$ **do**
        $q(B) \leftarrow q(B)/w(A)$;

References

[1] M. Chaveroche, F. Davoine, and V. Cherfaoui, "Efficient Möbius transformations and their applications to DS theory," in *International Conference on Scalable Uncertainty Management*. Springer, 2019, pp. 390–403.

[2] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[3] A. Dempster, "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 30, 1968.

[4] J. A. Barnett, "Computational Methods for a Mathematical Theory of Evidence," *Proc. of IJCAI*, vol. 81, pp. 868–875, 1981.

[5] J. Gordon and E. H. Shortliffe, "A method for managing evidential reasoning in a hierarchical hypothesis space," *Artificial intelligence*, vol. 26, no. 3, pp. 323–357, 1985.

[6] P. P. Shenoy and G. Shafer, "Propagating Belief Functions with Local Computations," *IEEE Expert*, vol. 1, no. 3, pp. 43–52, 1986.

[7] G. Shafer and R. Logan, "Implementing Dempster's rule for hierarchical evidence," *Artificial Intelligence*, vol. 33, no. 3, pp. 271–298, 1987.

[8] R. Kennes, "Computational aspects of the Mobius transformation of graphs," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 201–223, 1992.

[9] N. Wilson, "Algorithms for Dempster-Shafer Theory," in *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Algorithms for Uncertainty and Defeasible Reasoning*. Springer Netherlands, 2000, pp. 421–475.

[10] A. Sarabi-Jamab and B. N. Araabi, "Information-Based Evaluation of Approximation Methods in Dempster-Shafer Theory," *IJUFKS*, vol. 24, no. 04, pp. 503–535, 2016.

[11] M. Chaveroche, F. Davoine, and V. Cherfaoui, "Calcul exact de faible complexité des décompositions conjonctive et disjonctive pour la fusion d'information," in *Proceedings of XXVIIth Francophone Symposium on signal and image processing (GRETSI)*, 2019.

[12] ——, "Efficient exact computation of the conjunctive and disjunctive decompositions of D-S Theory for information fusion: Translation and extension," 2021. [Online]. Available: https://arxiv.org/abs/2107.06329

[13] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, "Trimmed Moebius inversion and graphs of bounded degree," *Theory of Computing Systems*, vol. 47, no. 3, pp. 637–654, 2010.

[14] M. Chaveroche, F. Davoine, and V. Cherfaoui, "Focal points and their implications for möbius transforms and dempster-shafer theory," *Information Sciences*, vol. 555, pp. 215 – 235, 2021.

[15] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, J. Nederlof, and P. Parviainen, "Fast zeta transforms for lattices with few irreducibles," *ACM TALG*, vol. 12, no. 1, p. 4, 2016.

[16] G.-C. Rota, "On the foundations of combinatorial theory I. Theory of Möbius functions," *Probability theory and related fields*, vol. 2, no. 4, pp. 340–368, 1964.

[17] P. Kaski, J. Kohonen, and T. Westerbäck, "Fast Möbius inversion in semimodular lattices and U-labelable posets," *arXiv preprint arXiv:1603.03889*, 2016.

[18] M. Chaveroche, "evidence-based-DST," https://github.com/mchaveroche/evidence-based-DST, 2019.

[19] R. Haenni and N. Lehmann, "Implementing belief function computations," *International Journal of Intelligent Systems*, vol. 18, no. 1, pp. 31–49, 2003.

[20] L. G. Polpitiya, K. Premaratne, M. N. Murthi, and D. Sarkar, "A Framework for efficient computation of belief theoretic operations," *Proc. of FUSION*, pp. 1570–1577, 2016.

[21] ——, "Efficient Computation of Belief Theoretic Conditionals," *Proc. of ISIPTA*, pp. 265–276, 2017.

[22] T. Denoeux, "Conjunctive and disjunctive combination of belief functions induced by nondistinct bodies of evidence," *Artificial Intelligence*, vol. 172, no. 2, pp. 234 – 264, 2008.