A Lattice Representation of Independence Relations

Linda C. van der Gaag L.C.VANDERGAAG@UU.NL
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Marco Baioletti MARCO.BAIOLETTI@UNIPG.IT
Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Italy

Janneke H. Bolt J.H.BOLT@UU.NL
Department of Information and Computing Sciences, Utrecht University, The Netherlands

Abstract

Independence relations in general include exponentially many statements of independence, that is, exponential in terms of the number of variables involved. These relations are typically fully characterised however, by a small set of such statements and an associated set of derivation rules. While various computational problems on independence relations can be solved by manipulating these smaller sets without the need to explicitly generate the full relation, existing algorithms are still associated with often prohibitively high running times. In this paper, we introduce a lattice representation for sets of independence statements, which provides further insights in the structural properties of independence and thereby renders the algorithms for some well-known problems on independence relations less demanding. By means of experimental results, in fact, we demonstrate a substantial gain in efficiency of closure computation of semi-graphoid independence relations.

Keywords: Independence relations; Representation; Lattice-based partitioning; Combinatorial complexity; Efficiency of closure computation.

1. Introduction

Probabilistic independence is the key to the scalability of probabilistic models, as is demonstrated by the practicability of probabilistic graphical models encoding such independences. Probabilistic independence therefore is a subject of intensive studies from both a mathematics and a computing-science perspective (see for example Dawid (1979); Pearl (1988); Studený (2005)). Pearl and his colleagues were among the first to formalise properties of probabilistic independence in a system of axioms (Pearl, 1988). These axioms are now commonly taken as derivation rules for generating new independences from a given set of independence statements. Any set of such statements that is closed under finite application of these rules is called an *independence relation*.

Independence relations in general are exponentially large in the number of variables involved. Representing them by enumeration of their element statements therefore is not feasible for practical purposes. These relations are typically fully characterised however, by a much smaller set of statements, called a *basis*, and an associated set of derivation rules. Studený was the first to propose a type of basis that allows various computational problems on independence relations to be solved efficiently without the need to generate the full relation (Studený, 1997; Studený, 1998). He designed an elegant algorithm for computing such a basis from a given starting set of independence statements, which was later improved upon, first by Baioletti et al. (2009a) and then by Lopatatzidis and van der Gaag (2017), to the current state-of-the-art algorithm for *closure computation*.

While Studený's basis allows efficiently solving various problems on independence relations, existing algorithms for finding such a basis are still highly demanding from a computational per-

spective. In this paper, we introduce a lattice representation for sets of independence statements which aims at reducing the running time of manipulating such sets through a partitioning approach. Although the lattice representation accommodates any type of independence relation, we will focus the discussion of its properties on aspects of closure computation of semi-graphoid independence relations. We will show that the lattice representation supports the maintenance of a non-redundant set of independence statements, which is one of the most demanding steps of closure computation, by restricting redundancy checks to parts of the lattice; the representation further allows effective selection of statements for application of the main operator involved in closure computation. Our experimental results from closure computation with up to 50 variables demonstrate that the computational advantages thus obtained are substantial.

The paper is organised as follows. In Section 2, we review some basic notions from independence relations and closure computation, and thereby introduce our notational conventions. In Section 3, our lattice representation of sets of independence statements is introduced and some of its properties are stated. Section 4 then details the computational advantages, for closure computation, of the new representation. Section 5 describes our experiments and the results obtained. The paper ends in Section 6 with our concluding observations and plans for further research.

2. Preliminaries

We briefly review semi-graphoid independence relations and their representation (Pearl, 1988; Studený, 1998). To this end, we consider a finite, non-empty set V of discrete random variables, with $|V| = n$, $n \geq 2$. A *triplet* over V is a statement of the form $\theta = \langle A, B \,|\, C \rangle$, where $A, B, C \subseteq V$ are pairwise disjoint subsets with $A, B \neq \varnothing$; the *symmetric transpose* of θ, denoted as θ^T, is the triplet $\langle B, A \mid C \rangle$. In the sequel, we will use $X = A \cup B \cup C$ to denote the set of all variables involved in the triplets θ, θ^T. A triplet $\langle A, B \mid C \rangle$ is taken to state that the sets of variables A and B are independent given the *separating set* C. Relative to a discrete joint probability distribution $\Pr$ over V, the triplet thus states that $\Pr(A, B \mid C) = \Pr(A \mid C) \cdot \Pr(B \mid C)$ for all value combinations of $A \cup B \cup C$. The set of all possible triplets over V is denoted by $V^{(3)}$.

A set of triplets constitutes a *semi-graphoid independence relation* if it satisfies the four properties stated in the following definition.

Definition 1 *A* semi-graphoid independence relation *is a set of triplets $J \subseteq V^{(3)}$ which satisfies the following properties:*

G1: *if* $\langle A, B \mid C \rangle \in J$, *then* $\langle B, A \mid C \rangle \in J$ (Symmetry)

G2: *if* $\langle A, B \mid C \rangle \in J$, *then* $\langle A, B' \mid C \rangle \in J$ *for any non-empty subset* $B' \subseteq B$ (Decomposition)

G3: *if* $\langle A, B_1 \cup B_2 \mid C \rangle \in J$ *with* $B_1 \cap B_2 = \varnothing$, *then* $\langle A, B_1 \mid C \cup B_2 \rangle \in J$ (Weak Union)

G4: *if* $\langle A, B \mid C \cup D \rangle \in J$ *and* $\langle A, C \mid D \rangle \in J$, *then* $\langle A, B \cup C \mid D \rangle \in J$ (Contraction)

The four properties stated above constitute an axiomatic system for the qualitative notion of independence (Pearl, 1988). This system is sound for the class of discrete probability distributions (Dawid, 1979), but not complete; in fact, it has been shown that the probabilistic notion of independence does not allow a finite axiomatisation (Studený, 1992).

The semi-graphoid properties of independence G1–G4 are now taken as derivation rules for generating (possibly) new triplets from a given triplet set. Given such a set $J \subseteq V^{(3)}$ and a designated triplet $\theta \in V^{(3)}$, we write $J \vdash^* \theta$ if the triplet θ can be derived from J by finite application

of the semi-graphoid rules. The (*full*) *closure* of a triplet set J, denoted as $\overline{J}$, is the semi-graphoid independence relation composed of J and all triplets θ that can be derived from it. Various computational problems on independence relations can be solved from a starting set J directly, using the derivation rules, without the need to first generate its full closure. An example of such a problem is the *implication problem* which asks, for a given starting set J and triplet θ, whether $J \vdash^* \theta$.

The semi-graphoid rules G1–G3 define a derivational partial order among triplets. Studený (1998) formalised this derivational order in the notion of *dominance*, which was later enhanced to that of *g-inclusion* by Baioletti et al. (2009a). In the sequel, we will use the latter notion.

Definition 2 *Let $J \subseteq V^{(3)}$ be a set of triplets and let $\theta_i = \langle A_i, B_i \,|\, C_i \rangle \in J$, with $X_i = A_i \cup B_i \cup C_i$, $i = 1, 2$. Then, θ_1 is* g-included *in θ_2, denoted as $\theta_1 \sqsubseteq \theta_2$, if the following conditions hold:*

- $C_2 \subseteq C_1 \subseteq X_2$*; and,*
- $[A_1 \subseteq A_2$ *and* $B_1 \subseteq B_2]$ *or* $[B_1 \subseteq A_2$ *and* $A_1 \subseteq B_2]$.

A triplet $\theta \in J$ is g-maximal *in J if it is not g-included in any triplet $\tau \in J$ with $\tau \neq \theta, \theta^T$.*

The conditions for g-inclusion capture all possible ways in which the triplet θ_1 may be derived from θ_2 by means of the derivation rules G1–G3. We note that, if a triplet θ and its symmetric transpose θ^T both are an element of a triplet set J, then g-maximality of θ in J implies g-maximality of θ^T in J, and vice versa. In the sequel, a set of g-maximal triplets which includes at most one of each pair of symmetric transposes, will be termed a *non-redundant triplet set*.

For application of the contraction rule G4, Studený (1998) formulated a dedicated operator which in essence constructs from two triplets θ_1, θ_2, two triplets θ_1', θ_2' with $\theta_1' \sqsubseteq \theta_1$ and $\theta_2' \sqsubseteq \theta_2$, to which G4 can be applied to yield a possibly new maximal triplet. This *gc-operator* is as follows.

Definition 3 *Let V be as above. For all triplets $\theta_i = \langle A_i, B_i \,|\, C_i \rangle \in V^{(3)}$ with $X_i = A_i \cup B_i \cup C_i, i = 1, 2$, such that*

- $A_1 \cap A_2 \neq \varnothing$*;*
- $C_1 \subseteq X_2$ *and* $C_2 \subseteq X_1$*; and*
- $(B_2 \setminus C_1) \cup (B_1 \cap X_2) \neq \varnothing$*,*

the gc-operator *is defined through:*

$$gc(\theta_1, \theta_2) = \langle A_1 \cap A_2,\ (B_2 \setminus C_1) \cup (B_1 \cap X_2) \ \mid\ C_1 \cup (A_1 \cap C_2) \rangle$$

For all triplet pairs θ_1, θ_2 for which the above conditions do not all hold, $gc(\theta_1, \theta_2)$ is undefined.

Usually, when a triplet pair θ_1, θ_2 is selected for application of the gc-operator, not just $gc(\theta_1, \theta_2)$ but also $gc(\theta_1^T, \theta_2)$, $gc(\theta_1, \theta_2^T)$, $gc(\theta_1^T, \theta_2^T)$ are established; in the sequel, we use $GC(\theta_1, \theta_2)$ to denote the set of all valid triplets resulting from this fourfold application of the operator. We note that for any triplet pair θ_1, θ_2, the set $GC(\theta_1, \theta_2) \cup GC(\theta_2, \theta_1)$ may include up to eight valid triplets.

For representing a semi-graphoid independence relation, it suffices to find a tailored subset of triplets, called a *basis*, that captures, jointly with the derivation rules, the same information as the entire relation itself. Studený (1998) was the first to design an algorithm for generating, from a

starting set of triplets, a tailored basis from which the implication problem allowed an efficient solution. This basis was composed of maximal triplets, constructed iteratively from a given starting set by applying the gc-operator and subsequently removing all g-included triplets. The algorithm was later enhanced, first by Baioletti et al. (2009a) and then by Lopatatzidis and van der Gaag (2017), to the state-of-the-art *algorithm for closure computation*.

3. A Lattice Representation of Triplet Sets

We introduce a lattice representation for arbitrary sets of triplets, and state some of its properties.

3.1 Partitioning a triplet set

We consider the power set of the set of variables V, and its representation in a lattice $\mathcal{L}$. As usual, each element of the lattice represents a subset of V and its links capture the $\subseteq$-relation between the sets; the element $\varnothing$ constitutes the meet of the lattice, and its join is the full set V. The lattice element representing the set $W \subseteq V$ will be denoted by $\mathcal{L}(W)$. We will further use $\mathcal{L}([W;X])$, with $W \subseteq X \subseteq V$, to denote the sublattice of $\mathcal{L}$ with W for its meet and X for its join. We now take this lattice $\mathcal{L}$ to represent a set J of triplets over V, by looking upon its elements as separating sets. More specifically, a lattice element $\mathcal{L}(C)$ represents the subset of all triplets $\theta = \langle A, B \,|\, C \,\rangle$ from J having the same separating set C. Since the separating set of a triplet cannot include more than $n-2$ elements, the semantics of the lattice does not allow any triplets at its two upper levels. In the sequel, we will use the term lattice therefore, to refer to a meet lattice with $n-1$ levels.

Through the lattice representation, an arbitrary triplet set J is partitioned into disjoint subsets associated with the elements of the lattice $\mathcal{L}$. The following lemma provides an upper bound on the cardinality of the triplet set represented by a single lattice element.

Lemma 4 *Let $\mathcal{L}$ be the lattice for the variable set V as described above. Then, for each subset $C \subseteq V$ with $|\,C\,| = n - m$, $n \geq m \geq 2$, the lattice element $\mathcal{L}(C)$ represents a set of at most $3^m - 2^{m+1} + 1$ different triplets.*

Proof The lattice element $\mathcal{L}(C)$ represents only triplets of the form $\theta_j = \langle A_j, B_j \,|\, C \rangle$ with at least 2 and at most m variables in $A_j \cup B_j$ jointly. For any fixed selection of $i \geq 2$ variables from among the m variables in $V \setminus C$, there exist $S(i,2) = 2^{i-1} - 1$ partitions into two subsets, with $S(i,2)$ the associated Stirling partition number. Each of these partitions defines an *unordered* pair of variable sets, and therefore has associated two symmetric triplets in which the two sets are ordered. As there are $\binom{m}{i}$ ways to choose exactly i variables from among a set of m, there are at most $2 \cdot \binom{m}{i} \cdot (2^{i-1}-1)$ triplets θ_j with $|A_j \cup B_j| = i$. By summing over all possible values $i = 2, \ldots, m$, the number of different triplets represented by the lattice element $\mathcal{L}(C)$ is found to be at most

$$2 \cdot \sum_{i=2}^{m} \binom{m}{i} \cdot (2^{i-1} - 1) \;=\; 3^m - 2^{m+1} + 1$$

as stated in the lemma. ■

In the proof above, we used the Stirling partition number to indicate the number of possible partitions of a fixed selection of i variables into two non-empty subsets, where each such partition gives rise to two symmetric triplets. The number stated in the lemma thereby is an upper bound on the

overall number of triplets that can be represented by a lattice element. In fact, this upper bound can be attained only if all corresponding symmetric and g-included triplets are also included in the representation. When manipulating independence relations however, typically a non-redundant triplet set is maintained, without any g-included or symmetric triplets, as such triplets do not carry any additional information. The number of triplets of the largest possible non-redundant triplet set represented by a lattice element is considerably smaller than the number stated above. We will presently conjecture an upper bound on the cardinality of this largest possible set. To support our conjecture, we present two lemmas, in which we will consider the sets J_C^i, for a lattice element $\mathcal{L}(C)$, including all possible triplets with exactly i variables in their first two arguments jointly, yet *without symmetric transposes*; we note that such a set includes $\binom{m}{i} \cdot (2^{i-1} - 1)$ triplets. Our first lemma now states that, for any fixed number i, the set J_C^i is a non-redundant triplet set.

Lemma 5 *Let V be as before and let $C \subseteq V$ with $|C| = n - m$, $n \geq m \geq 2$. Let $J_C^i, i = 2, \ldots, m$, be as described above. Then, for all $i = 2, \ldots, m$, J_C^i is a non-redundant triplet set.*

Proof For any ordered pair of different triplets θ_1, θ_2 chosen from the set J_C^i for some i, we have to show that $\theta_1 \not\sqsubseteq \theta_2$. Since by the construction of J_C^i we have that $|A_1 \cup B_1| = |A_2 \cup B_2|$, we find for all θ_1, θ_2 with $|A_1| \neq |A_2|$ that the conditions $A_1 \subseteq A_2$ and $B_1 \subseteq B_2$ cannot both hold; for all θ_1, θ_2 with $|A_1| = |A_2|$ and, hence, $|B_1| = |B_2|$, moreover, we find that if $A_1 = A_2$ then $B_1 \neq B_2$, and vice versa, from which we equally have that the conditions $A_1 \subseteq A_2$ and $B_1 \subseteq B_2$ cannot both hold. As similar observations hold with θ_1^T, we thus find that $\theta_1 \not\sqsubseteq \theta_2$ for all ordered pairs $\theta_1, \theta_2 \in J_C^i$. We conclude that J_C^i is a non-redundant triplet set. ■

From Lemma 5 we have that each set J_C^i, $i = 2, \ldots, m$, is a non-redundant triplet set. The following lemma now indicates the number k which maximises cardinality among these sets J_C^i.

Lemma 6 *Let V be as before and let $C \subseteq V$ with $|C| = n - m$, $n \geq m > 3$. Let J_C^i, $i = 2, \ldots, m$, be as above. Then, for $k = \left\lfloor \frac{2}{3}m + \frac{2}{3} \right\rfloor$, we have that $|J_C^k| \geq |J_C^i|$ for all $i = 2, \ldots, m$.*

Proof For the cardinalities of any two consecutive triplet sets J_C^i and J_C^{i+1}, we have that

$$|J_C^{i+1}| = \left(\frac{m-i}{i+1} \cdot \frac{2^i - 1}{2^{i-1} - 1} \right) \cdot |J_C^i|$$

that is, the two sets differ in size by the factor $f_m(i) = \left(\frac{m-i}{i+1} \cdot \frac{2^i-1}{2^{i-1}-1} \right)$, $i = 2, \ldots, m-1$. For any number of variables $m > 3$, the function $f_m(i)$ decreases monotonically in i. To gain further insight in $f_m(i)$, we observe that

$$\frac{m-i}{i+1} \cdot \frac{2^i - 1}{2^{i-1} - 1} > 2 \cdot \frac{m-i}{i+1}$$

for $i = 2, \ldots, m-1$. As the term $2 \cdot \frac{m-i}{i+1}$ attains the value 1 at $i_m = \frac{2}{3}m - \frac{1}{3}$, we conclude, by its property of monotonic decrease, that the function $f_m(i)$ attains the value 1 at some real number $\hat{\imath}_m > i_m$. By some algebraic manipulation, we further find that

$$i_m < \hat{\imath}_m < i_m + \frac{1}{3} \cdot \left(\frac{m - i_m}{2^{i_m - 1} - 1} \right) < i_m + 1$$

for all $m > 3$. The smallest integer k larger than $\hat{\imath}_m$ now maximises cardinality among the sets J^i_C. To establish k, we distinguish between three forms of the number m. For an m of the form $m = 3x$ with $x \geq 2$ an integer, we find, with $\lceil i_m \rceil = \lfloor i_m + 1 \rfloor = \frac{2}{3}m$, that $f_m(\lfloor i_m + 1 \rfloor) < 1$; for an m of the form $m = 3x+1$ with $x \geq 1$ an integer, we equally find, with $\lceil i_m \rceil = \lfloor i_m + 1 \rfloor = \frac{2}{3}m + \frac{1}{3}$, that $f_m(\lfloor i_m + 1 \rfloor) < 1$. For these two forms, we thus have that $\lceil i_m \rceil = \lfloor i_m + 1 \rfloor$ is the smallest integer larger than $\hat{\imath}_m$. For an m of the form $m = 3x + 2$ with $x \geq 1$ an integer, we find that $i_m + 1$ itself is an integer and, hence, that $i_m + 1 = \lfloor i_m + 1 \rfloor$ is the smallest integer larger than $\hat{\imath}_m$. We conclude that $k = \lfloor i_m + 1 \rfloor = \lfloor \frac{2}{3}m + \frac{2}{3} \rfloor$ maximises cardinality among the sets J^i_C for $i = 2, \ldots, m$. ■

Lemmas 5 and 6 jointly show that the largest possible non-redundant triplet set among the sets J^i_C is the set J^k_C with $k = \lfloor \frac{2}{3}m + \frac{2}{3} \rfloor$. Now suppose that there exists a non-redundant triplet set of larger cardinality than this set J^k_C, that can be represented by the lattice element $\mathcal{L}(C)$. It is readily shown that each triplet from the set $J^{k+1}_C \cup \ldots \cup J^m_C$ g-includes at least one triplet from J^k_C; similarly, each triplet from the set $J^2_C \cup \ldots \cup J^{k-1}_C$ is g-included in at least one triplet from J^k_C. From these observations, we conclude that we cannot construct such a larger triplet set by simply adding triplets to J^k_C. Although this conclusion does not guarantee that no larger set exists, we cautiously conjecture that it is not possible to construct a non-redundant triplet set of larger cardinality than $|J^k_C|$ for representation at the lattice element $\mathcal{L}(C)$ and, hence, that $\binom{m}{k} \cdot (2^{k-1} - 1)$ is a tight upper bound on the size of any non-redundant triplet set represented at this element.

3.2 Dynamically maintaining the triplet set representation

For deriving new triplets from a given triplet set upon closure computation, the gc-operator is applied to pairs of triplets from this set. If application of the operator to a triplet pair θ_1, θ_2 yields a valid triplet, the result $\theta = gc(\theta_1, \theta_2)$ is included in the overall triplet set and, hence, is inserted in the set's representation. The lattice element at which θ is inserted, is determined to a large extent by the elements from which θ_1, θ_2 were taken. We distinguish between two situations:

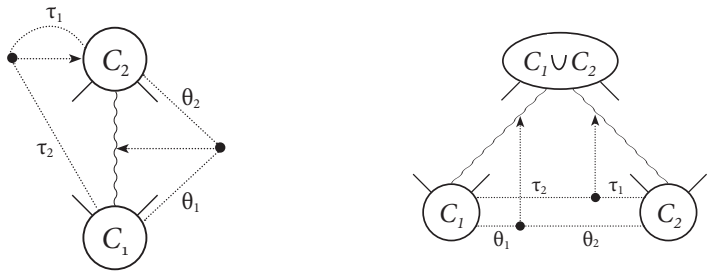
- We consider two lattice elements $\mathcal{L}(C_1), \mathcal{L}(C_2)$ with $C_1 \subseteq C_2$:
  - if valid, $\theta = gc(\theta_1, \theta_2)$, with $\theta_1 \in \mathcal{L}(C_1)$, $\theta_2 \in \mathcal{L}(C_2)$, is inserted in a lattice element in the sublattice $\mathcal{L}([C_1; C_2])$;
  - if valid, $\theta = gc(\tau_1, \tau_2)$, with $\tau_1 \in \mathcal{L}(C_2)$, $\tau_2 \in \mathcal{L}(C_1)$, is inserted in $\mathcal{L}(C_2)$.

  The two cases are illustrated in Figure 1(a). In the case where θ_1, θ_2 are both taken from the same lattice element, then, if valid, the result $gc(\theta_1, \theta_2)$ is inserted in this element.

- We consider two lattice elements $\mathcal{L}(C_1), \mathcal{L}(C_2)$ with $C_1 \setminus C_2 \neq \varnothing$, $C_2 \setminus C_1 \neq \varnothing$:
  - if valid, $\theta = gc(\theta_1, \theta_2)$, with $\theta_1 \in \mathcal{L}(C_1)$, $\theta_2 \in \mathcal{L}(C_2)$, is inserted in an element in the sublattice $\mathcal{L}([C_1; C_1 \cup C_2])$.
  - if valid, $\theta = gc(\tau_1, \tau_2)$, with $\tau_1 \in \mathcal{L}(C_2)$, $\tau_2 \in \mathcal{L}(C_1)$, is inserted in $\mathcal{L}([C_2; C_1 \cup C_2])$.

  These two cases are illustrated in Figure 1(b).

The dynamics of the triplet-set representation occasioned by the gc-operator, reveal that newly derived triplets cannot be inserted arbitrarily high (low) in the lattice; in fact, the level at which a new triplet is inserted is indirectly constrained by the highest (lowest) level of non-empty elements.



(a) (b)

Figure 1: The dynamics of inserting a valid result from applying the gc-operator, to a pair of triplets taken from elements on a direct path from the meet to the join of the lattice (a), and to a pair of triplets from lattice elements for which no such path exists (b), respectively.

4. Computational Advantages for Closure Computation

The lattice representation for sets of triplets introduced above, brings computational advantages for various problems on independence relations. In this section, we discuss some of these advantages for closure computation of a semi-graphoid independence relation.

4.1 Selecting triplets for application of the GC-operator

At the core of the state-of-the-art algorithm for closure computation for semi-graphoid independence relations, lies application of the GC-operator to all suitable pairs of triplets in the triplet set at hand. Given some triplet θ, selecting appropriately paired triplets θ' in essence involves checking all other triplets in this set. When the set is maintained as a simple list therefore, selecting all suitable pairs takes quadratic time in the number of triplets involved. Experimental results have shown that, particularly in intermediate iterations of the algorithm, the latter number can be several orders of magnitude larger than the size of the final basis (see for example (Baioletti et al., 2009b)).

We now address selecting suitable pairs of triplets for application of the GC-operator in the lattice representation of a triplet set.

Lemma 7 *Let $\mathcal{L}$ be the lattice for the variable set V as before. Let $J \subseteq V^{(3)}$ be a set of triplets partitioned over $\mathcal{L}$, and let $\theta = \langle A, B \,|\, C\rangle \in J$ with $X = A \cup B \cup C$. Then, $GC(\theta, \theta_i) = \varnothing$ for all triplets $\theta_i \in \mathcal{L}(C_i)$ with $C_i \setminus X \neq \varnothing$.*

Proof The property stated in the lemma follows immediately from the condition $C_i \subseteq X$ for application of the gc-operator to yield a valid triplet. ■

From Lemma 7 we have, given a triplet θ, that searching for appropriately paired triplets for application of the GC-operator can be restricted to the sublattice $\mathcal{L}([\varnothing; X])$. The following lemma shows that the search in this sublattice can be even further restricted.

Lemma 8 *Let $\mathcal{L}$, V and J be as above, and let $\theta = \langle A, B \,|\, C\rangle \in J$. Then,*

- *for all triplets $\theta_i \in \mathcal{L}(C_i)$ with $A \subseteq C_i$, $gc(\theta, \theta_i)$ and $gc(\theta, \theta_i^T)$ are undefined;*
- *for all triplets $\theta_i \in \mathcal{L}(C_i)$ with $B \subseteq C_i$, $gc(\theta^T, \theta_i)$ and $gc(\theta^T, \theta_i^T)$ are undefined;*
- *for all triplets $\theta_i \in \mathcal{L}(C_i)$ with $A \cup B \subseteq C_i$, $GC(\theta, \theta_i) = \varnothing$.*

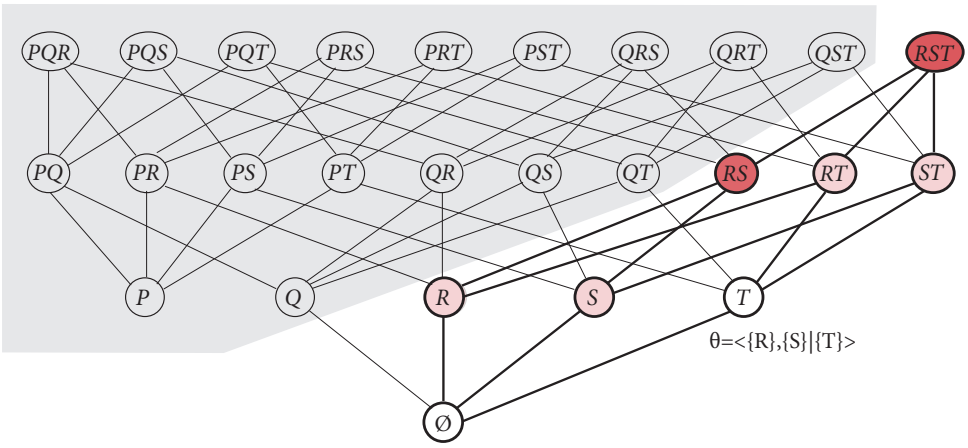


Figure 2: A lattice over five variables, illustrating the effects of Lemmas 7 and 8 on the selection of triplets for application of the GC-operator.

Proof The first property stated in the lemma follows from the condition $A \cap A_i \neq \varnothing$ for $gc(\theta, \theta_i)$ to be valid and the observation that $A \subseteq C_i$ and $A_i \cap C_i = \varnothing$ cause this condition to fail. Similar arguments hold for the other properties stated in the lemma. ■

We illustrate the effects of the two lemmas above on the selection of appropriately paired triplets by means of a small example.

Example 1 We consider the variable set $\{P, Q, R, S, T\}$ and the triplet $\theta = \langle\{R\}, \{S\} \mid \{T\}\rangle$. The lattice associated with the variable set is shown in Figure 2. When looking for triplets θ_i which are appropriately paired with θ for application of the GC-operator, we have from Lemma 7 that the search can be restricted to the sublattice $\mathcal{L}([\varnothing; \{R, S, T\}])$. The part of the lattice where appropriate triplets θ_i cannot reside by this lemma, is greyed out in the figure. From Lemma 8, we now further have that the lattice elements $\mathcal{L}(C_i)$ with $\{R, S\} \subseteq C_i$ also cannot contain any suitable triplets; these elements are indicated in dark grey in the figure. Searching for appropriately paired triplets θ_i therefore is restricted to just six lattice elements, out of the 26 elements of the lattice at large. From Lemma 8, we further have that, for the remaining lattice elements $\mathcal{L}(C_i)$ with either $R \in C_i$ or $S \in C_i$, two of the four possible applications of the gc-operator cannot yield a valid triplet; the lattice elements for which this property holds, are shown in light grey. All in all, the search for appropriately paired triplets θ_i for θ is effectively restricted to two lattice elements in full and to half of the applications of the gc-operator for four lattice elements. □

4.2 Checking for g-inclusion

The basic algorithm for closure computation of a semi-graphoid independence relation in essence is an iterative procedure in which each iteration involves applying the GC-operator to all appropriate triplet pairs and subsequently removing all g-included triplets. Checking for g-inclusion is a computationally demanding step of the algorithm. When maintaining a simple list of triplets for example,

```
1:  while J is changed do
2:    J' ← J
3:    for all θ ∈ J' do
4:      for all τ ∈ CANDIDATES(θ, J') do
5:        for all σ ∈ GC(θ, τ) do
6:          ADD(J, σ)
7:        end for
8:      end for
9:    end for
10: end while
```

Algorithm 1: An outline of the LFC algorithm.

the check for g-inclusion takes quadratic time in the number of triplets stored, per iteration. We now address checking for g-inclusion in the lattice representation of a triplet set.

Lemma 9 *Let $\mathcal{L}$ be the lattice for the variable set V as before. Let $J \subseteq V^{(3)}$ be a set of triplets partitioned over $\mathcal{L}$. Then, a triplet $\theta \in \mathcal{L}(C)$ can be g-included only in triplets $\theta' \in \mathcal{L}([\varnothing; C])$ and can only g-include triplets $\theta'' \in \mathcal{L}([C; X])$.*

Proof The property stated in the lemma follows immediately from Definition 2. ■

From the lemma we have that, for checking g-inclusion of a triplet $\theta \in \mathcal{L}(C)$, it needs to be compared only against triplets in the sublattice with $\mathcal{L}(C)$ for its join; for checking g-inclusion of other triplets by θ, it needs to be compared against triplets in the sublattice with $\mathcal{L}(C)$ for its meet and $\mathcal{L}(X)$ for its join. Since the notion of g-inclusion is transitive moreover, when maintaining a non-redundant triplet set, a new triplet can either be g-included in an existing triplet or itself g-include an existing triplet, but not both. We note that, despite these insights, even with the lattice representation, the number of triplet comparisons to be performed for checking g-inclusion remains to be quadratic in the worst case. In practice however, the runtime complexity will be smaller as each triplet is compared against only a subset of the overall triplet set.

5. Experiments and Results

To investigate the practical advantages of our lattice representation, we conducted various experiments in which we compared two implementations of the state-of-the-art closure algorithm for semi-graphoid independence relations: an implementation using a straightforward representation of triplet sets, as described in (Baioletti et al., 2009a), and an implementation using the lattice representation. We will refer to these implementations as the *FC algorithm* and the *LFC algorithm*, respectively. Algorithm 1 provides an outline of the LFC algorithm. The procedure CANDIDATES selects, given a triplet θ, all appropriately paired triplets τ from the current triplet set, for combination with θ through the GC-operator; this procedure extends the analogous procedure of the FC algorithm by incorporating the results from Section 4.1. After applying the GC-operator to a selected pair, the procedure ADD updates the lattice representation of the triplet set by inserting the resulting triplet σ, if valid, as described in Section 4.2, and then removing all g-included triplets.

For our experiments, we implemented an instance generator taking four parameters: the number of variables n, the number of starting triplets t, and two probabilities p and q with $2p + q < 1$. In a

Table 1: Experimental results for the running times and the numbers of performed g-inclusion checks of the LFC and FC algorithms respectively, for instances with $p = 0.2, q = 0.3$.

Instance		*LFC*			*FC*			*Summary* (in %)	
n	t	S_1	$\bar{T}_1$	$\bar{G}_1$	S_2	$\bar{T}_2$	$\bar{G}_2$	$\bar{T}_1/\bar{T}_2$	$\bar{G}_1/\bar{G}_2$
10	15	100	0.70	2267844.16	100	0.93	12204371.29	75.27	18.58
10	20	100	6.33	59742221.01	100	17.55	273741473.83	36.05	21.82
10	30	100	64.75	647100875.69	95	130.22	1897918808.48	49.72	34.10
10	50	99	237.36	2536240465.56	88	232.14	3356908842.64	102.25	75.55
20	30	99	1.63	2260410.09	98	3.16	49708165.60	51.64	4.55
20	40	95	8.03	36681653.78	91	37.22	629251618.65	21.57	5.83
20	60	38	13.16	60062590.63	35	78.28	1271880674.23	16.81	4.72
30	45	100	0.02	8370.20	100	0.03	248389.35	76.22	3.37
30	60	100	5.43	9882952.02	100	11.67	155335545.76	46.58	6.36
30	90	100	1.32	6097019.38	100	12.77	220879562.48	10.36	2.76
30	150	94	17.74	30758827.24	90	83.71	1338505451.08	21.19	2.30
40	60	100	0.01	2616.00	100	0.02	114748.28	40.36	2.28
40	80	100	0.01	1773.40	100	0.03	130964.09	28.23	1.35
40	120	100	0.03	41345.80	100	0.14	1510101.61	21.85	2.74
40	200	100	0.21	426506.96	100	1.12	14126751.07	18.53	3.02
50	75	100	0.00	98.87	100	0.01	12806.44	8.34	0.77
50	100	100	0.00	279.38	100	0.02	35329.40	9.28	0.79
50	150	100	0.01	705.65	100	0.05	118060.48	11.59	0.60
50	250	100	0.02	4709.39	100	0.16	639703.24	11.29	0.74

triplet $\theta = \langle A, B \,|\, C \rangle$ generated for an instance, each variable V_i occurs in the set A with probability p, in B with probability p, and in the separating set C with probability q; with probability $1-2p-q$, it does not appear in θ. We used the two probability parameters to compare the performances of the two algorithms on both simple and harder instances. The experiments were conducted with $n = 10, 20, 30, 40, 50$ variables and sets of $t = \lfloor 1.5n \rfloor, 2n, 3n, 5n$ triplets; for each value of n and associated value of t, we generated 100 triplet sets. We chose to set $p = 0.2$ and $q = 0.3$, to generate triplets in which the separating set C on average is larger in size than the sets A and B; by doing so, we feel that the generated triplets more closely match those found in applications. In order to complete the experiments in a reasonable amount of time, we set a time limit of 1200 seconds: if for a generated instance a basis was not returned within this time limit, its computation was halted.

Our first set of experiments were performed to gain insight in the extent to which the lattice representation helps reduce the computational burden of closure computation. To this end, we compared the running times and the numbers of g-inclusion checks performed by the two algorithms. Our results are summarised in Table 1. The columns S_1, $\bar{T}_1$ and $\bar{G}_1$ in the table report the number of instances on which computations were completed within the time limit, the average execution time in seconds, and the average number of g-inclusion checks performed by the LFC algorithm, respectively; the columns S_2, $\bar{T}_2$, and $\bar{G}_2$ report similar statistics for the FC algorithm. The columns $\bar{T}_1/\bar{T}_2$ and $\bar{G}_1/\bar{G}_2$ report the percentage ratios of $\bar{T}_1$ and $\bar{T}_2$, and of $\bar{G}_1$ and $\bar{G}_2$, respectively.

Table 2: The dynamics of the lattice during a run of the LFC algorithm, with $n = 10$, $t = 30$.

i	$\|J\|$	*GC generated*	$\bar{H}$	*max H*	$\|\mathcal{L}\|$	*min size*	*avg size*	*max size*
1	130	192	3.58	6	45	1	2.89	17
2	348	4453	3.36	7	67	1	5.19	59
3	640	45784	3.06	6	72	1	8.89	155
4	855	225462	2.86	6	48	1	17.81	200
5	1088	520074	2.87	6	31	1	35.10	216
6	1199	729152	2.82	6	24	1	49.96	257
7	1148	727846	2.78	4	19	1	60.42	245
8	1172	333787	2.81	4	18	1	65.11	233

Table 1 shows that the LFC algorithm was faster than the FC algorithm with each parameter combination n, t, except $n = 10$, $t = 50$. With this particular parameter combination, the triplets were often concentrated in a single lattice element, which effectively forestalled the advantages from the lattice representation. The (small) increase in running time may be attributed to the overhead of maintenance of the lattice representation and/or to implementational differences of the two algorithms with respect to code optimisation. Averaged over all instances solved by both algorithms, the LFC algorithm proved 3.3 times faster than the FC algorithm. The associated percentage ratios range from LFC taking some 8% of the running time of FC (with $n = 50, t = 75$), to LFC taking around 76% of FC's running time (with $n = 30, m = 45$). We note that, for each value of n, as t increases, the average running times of both algorithms increase. The running times do not exactly increase with n, however. All instances with $n \geq 30$ were easy to solve for both algorithms, while the hardest instances were those with $n = 10, t = 50$ and those with $n = 20$; the instance with the parameter combination $n = 20, t = 100$ was even excluded from the table as both algorithms were able to solve just a single instance within the time limit. Table 1 further shows that the LFC algorithm performed fewer g-inclusion checks than the FC algorithm with each parameter combination n, t. Averaged over all instances solved by both algorithms, the LFC algorithm in fact performed just 26% of the number of g-inclusion checks performed by FC.

A second experiment was conducted to illustrate the dynamics of the lattice during closure computation for a semi-graphoid independence relation. Table 2 reports some characteristics of the lattice, from a single run of the LFC algorithm on a representative instance with $n = 10$, $t = 30$. The column indicated by i states the iteration number; $|J|$ reports the size of the (non-redundant) triplet set J with which the iteration started and *GC generated* states the number of triplets generated by the GC-operator during the iteration; the columns $\bar{H}$ and *max H* state the average and maximum height of the non-empty part of the lattice; the four final columns report the number of non-empty lattice elements and their minimum, average and maximum sizes, respectively. The table shows that the size of the triplet set J and the number of generated triplets per iteration initially increase considerably, to more or less decrease towards the end of the computation. The maximum and average heights of the lattice's non-empty part tend to somewhat decrease per iteration, while the number of non-empty lattice elements strongly decreases after an initial increase. The average and maximum numbers of triplets per lattice element further increase substantially per iteration, with the maximum number decreasing at the end of the computation. These findings indicate that triplets tend to become concentrated in the lower part of the lattice as the closure computation progresses.

6. Concluding observations

Despite important advances in the past decades, manipulating independence relations still is computationally challenging, not just for the worst instances. Through a lattice representation of sets of independence statements, we reduced the running times involved in closure computation in practice. We showed that our representation supports maintaining a non-redundant set of independence statements by allowing redundancy checks to be restricted to parts of the lattice and that the representation further provides effective selection of statements for application of the main operator involved. Through experimental results, the computational advantages of using the lattice representation were demonstrated to be substantial.

Having studied the advantages of our lattice representation for semi-graphoid independence relations, we are now extending our investigations to other types of independence relation, among which are the graphoid relations. We are also expanding our experimental studies, focusing on the relation between the parameters involved and the hardness of instances. Since our study of the lattice representation uncovered some structural regularities of independence, we plan to further investigate these regularities from a foundational perspective, with the ultimate goal of arriving at algorithms for manipulating independence relations with improved properties of scalability.

References

M. Baioletti, G. Busanello, and B. Vantaggi. Conditional independence structure and its closure: inferential rules and algorithms. *International Journal of Approximate Reasoning*, 50:1097–1114, 2009a.

M. Baioletti, G. Busanello, and B. Vantaggi. Closure of independencies under graphoid properties: some experimental results. In *International Symposium on Imprecise Probability: Theories and Applications*, pages 11–19, Durham, 2009b.

A. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society B*, 41:1–31, 1979.

S. Lopatatzidis and L. C. van der Gaag. Concise representations and construction algorithms for semi-graphoid independency models. *International Journal of Approximate Reasoning*, 80:377–392, 2017.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

M. Studený. Conditional independence relations have no finite complete characterization. In *Information Theory, Statistical Decision Functions and Random Processes*, pages 377–396, Kluwer, Dordrecht, 1992.

M. Studený. Semigraphoids and structures of probabilistic conditional independence. *Annals of Mathematics and Artificial Intelligence*, 21:71–98, 1997.

M. Studený. Complexity of structural models. In *Proceedings of the Joint Session of the 6th Prague Conference on Asymptotic Statistics and the 13th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pages 521–528, Prague, 1998.

M. Studený. *Probabilistic Conditional Independence Structures*. Springer Verlag, London, 2005.