TITLE PAGE ICCS'96

Title:	LATTICE EMBEDDING
Author:	J.J. Sarbo
Address:	University of Nijmegen Toernooiveld 1, 6525 ED Nijmegen The Netherlands
Tel.nr.:	31-024-3653049
Fax nr.:	31-024-3553450
e-mail:	janos@cs.kun.nl

Abstract

Hierarchical representation of objects using lattices is one of the fundamental techniques in knowledge representation. In many applications of such representations e.g. conceptual combination, inheritance lattices and formal concept analysis, part of the computational problem can be resolved by lattice embedding.

Lattices can be defined in terms of different notions. We will consider a lattice as defined by a partially ordered set and its graph(ical) representation, a Hasse diagram. A lattice can equivalently be defined as a directed graph. Since embedding is a subcase of the more general notion of homomorphism, we will first investigate lattice homomorphism with respect to related problems on graphs, like graph isomorphism, (sub)graph homomorphism and (sub)graph homeomorphism, as a preliminary to our problem.

We develop an algorithm for lattice embedding (which is NP-complete) and prove its correctness and termination. Finally we show how the complexity of the algorithm can be lowered.

Lattice Embedding

Janos J. Sarbo

University of Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract. In this paper we investigate lattice homomorphism and in particular the embedding problem for lattices which is NP-complete. The use of lattice embedding in knowledge representation is showed by an example.

1 Introduction

Hierarchical representation of objects using lattices is one of the fundamental techniques in knowledge representation. In many applications of such representations e.g. conceptual combination ([CK92]), inheritance lattices ([GM93]) and formal concept analysis ([Wil82, SF95]), part of the computational problem can be resolved by lattice embedding. This is exemplified in section 5 of the paper.

Lattices can be defined in terms of different notions. We will consider a lattice as defined by a partially ordered set and its graph(ical) representation, a Hasse diagram. A lattice can equivalently be defined as a directed graph. Since lattice embedding is a subcase of the more general notion of homomorphism, we will first investigate lattice homomorphism with respect to related problems on graphs, like graph isomorphism, (sub)graph homomorphism and (sub)graph homeomorphism, as a preliminary to our problem.

The embedding problem for lattices considered as graphs is not known in the literature (at least we could not find references). More is known about the homomorphism of lattices considered as algebraic structures, but these results do not seem directly applicable to our lattice embedding problem.

2 Basic definitions

A *partial order* on a set P is a binary relation $\leq$ which is reflexive, anti-symmetric and transitive. A partially ordered set will also be called a *poset*. Usually we shall be a little slovenly and say P is a partially ordered set, or $x \in P$ for an element in P. If two elements x, y are ordered $x \leq y$ then x is *contained* in or *smaller* than y. If $x < y$ and $x \leq z < y$ implies $x = z$ then x is *covered* by y (or y *covers* x). A traditional representation for a partial order is by its *Hasse diagram* ([DP90]).

A *directed graph* or *digraph* is a pair $G = (V, E)$ where V is a set and E a set of ordered pairs from V. Elements of V and E are called nodes or vertices, and edges or arcs, resp. An edge (x, y) *originates* at x and *terminates* at y; x and y are called its *end* vertices. The number of arcs which originate at a node x is called the *outdegree* of node x; the number of arcs which terminate at y is called the *indegree* of node y. A node x is *adjacent* with a node y if there is an edge (x, y) or an edge (y, x). We say graph G_1 is a *subgraph* of a graph G_2 if every node and edge of G_1 is included in G_2 (G_2 is called a *supergraph* of G_1).

A *directed path* p from node x to z is a finite sequence of nodes $p = y_1 y_2 \ldots y_n$ such that $y_1 = x$, $y_n = z$, and for all y_i, $1 \leq i < n$, (y_i, y_{i+1}) is an edge. The length of a path is denoted by *length*(p) or $|p|$. If for a path $y_1 = y_n$ $(1 < n)$ then it is called a *cycle*. A cycle-free path is called a *simple path*. A directed graph, not containing any cycle, is called a *directed acyclic graph*. A set of paths U is called *pairwise node disjoint* if any two distinct paths from U are disjoint, that is they do not have common nodes except possibly for their first and last nodes. We say a node x *reaches* node y (y *is reachable* from x) if there is a path from x to y. If L is a finite poset and $x \in L$, then $height(x)$ (resp. $depth(x)$) is the maximal cardinality of a path $xv_1 \ldots v_n \top$ (resp. $\bot v_1 \ldots v_n x$) in L.

Traditionally, a *lattice* is defined as a poset ([Bir67]) in which any two elements have a meet ($\sqcap$) or greatest lower bound and a join ($\sqcup$) or least upper bound. For a lattice $L = (V, \leq)$ the corresponding graph $G = (V, E)$ is such that for $a, b \in V$: $(a, b) \in E$ if a covers b. As we are only interested in finite lattices, in the rest of the paper a lattice will always denote a finite one. The following lemma is taken from [Bir67].

Lemma 1. *For a lattice $L = (V, E)$ the collection of subsets $C_i \subseteq V$ with $C_i = \{x \in V \mid depth(x) = i\}$, for all depths in L, is a partitioning of V.*

We introduce subsets of a lattice for an element, collecting all elements the given element contains or is contained by.

Definition 2. Let L be a lattice, the elements of L ordered by $\leq$ and $x \in L$. We define $\triangle(x)$, the up-set of x as $\triangle(x) = \{y \in L \mid x \leq y\}$; and $\triangledown(x)$, the down-set of x as $\triangledown(x) = \{y \in L \mid y \leq x\}$.

Definition 3. A *sublattice* of a lattice L is a subset X of L such that $x \in X$, $y \in X$ imply $x \sqcap y \in X$ and $x \sqcup y \in X$.

Lemma 4. *Let L be a lattice and $x \in L$. Then $\triangledown(x)$ and $\triangle(x)$ are sublattices of L.*

In the theory of finite lattices *irreducible* elements play a role analogue of primes in the arithmetics ([Wil77]).

Definition 5. A node x of a lattice L, with $x \neq \bot$ and $x \neq \top$ is called $\sqcup$-*irreducible* (resp. $\sqcap$-*irreducible*) when $x = y \sqcup z$ (resp. $x = y \sqcap z$) implies $x = y$ or $x = z$.

If only the conditions, either for the meets or joins have to be satisfied we obtain *semilattices*.

3 Towards lattice embedding

The embedding problem for lattices is concerned with the question whether a *test* (or input) lattice (L_t) embeds the structure of a *pattern* lattice (L_p). As we may consider lattices as directed graphs or graphs of partially ordered sets, we will describe lattice embedding from a graph theoretical point of view.

3.1 Lattice isomorphism

The isomorphism problem for lattices is concerned with finding all isomorphic mappings between two lattices. A definition of this problem is based on one for graphs. We will use the terms *mapping*, *match* and *matching* to denote injective functions.

Definition 6. Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be directed graphs. Are G and H isomorphic, i.e. is $|V_G| = |V_H|$ and $|E_G| = |E_H|$, and is there a mapping φ from G to H such that for all $x, y \in V_G$:

$$(x, y) \in E_G \Leftrightarrow (\varphi(x), \varphi(y)) \in E_H \ ? \tag{1}$$

The related problem for lattices is formulated as follows.

Definition 7. Let L_t and L_p be finite lattices. Is there a mapping φ from L_p to L_t such that φ is an isomorphism from L_p to $\varphi(L_p)$?

We prove by example that there may be exponentially many such isomorphic mappings.

Example 1. Consider the Hasse diagram of M_n in Fig. 1. For M_n and M_m ($m \leq n$) a mapping from M_m to M_n must project top and bottom of M_m onto those of M_n since the mapping has to preserve adjacency. The remaining $m-2$ elements of M_m can freely be mapped to the remaining $n-2$ elements of M_n resulting in $(n-2)!/(n-m)!$ possible mappings from M_m to M_n. As we are interested in isomorphisms, $n = m$ and the number of isomorphic mappings is $(n-2)!$ which is exponential in n.

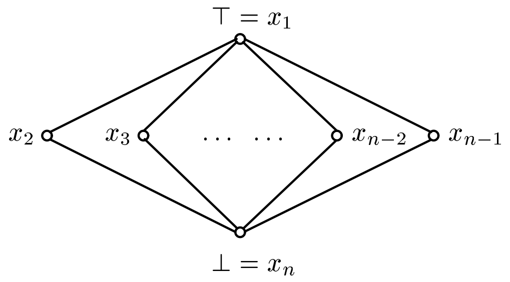

Fig. 1. The Hasse diagram of M_n

3.2 Lattice homeomorphism

Homeomorphism is probably best known as the topological notion of an isomorphism. A homeomorphism on graphs is a property that, among others, preserves planarity. The homeomorphism property can be incorporated in an embedding problem, the subgraph homeomorphism problem. This problem is known to be of polynomial complexity for ordered trees and NP-complete for general graphs [FKMP95]. The following definition is for graphs, a definition for lattices can be similarly given.

Definition 8. Let G and H be directed graphs. G is *homeomorphic* to H if H is isomorphic to a graph obtained from G by repeatedly removing any node v of in- and outdegree 1 and adding the arc (u, w), where (u, v) and (v, w) were in G. A graph G is *subgraph homeomorphic* to H if G has a subgraph I homeomorphic to H.

3.3 Lattice homomorphism

The following definition for undirected graph homomorphism is taken from [Joh82].

Definition 9. Two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, with self-loops (i.e. edges of the form (x, x) allowed) are homomorphic from H to a subgraph of G if there is a function φ from V_H to V_G such that if $(u, v) \in E_H$ then $(\varphi(u), \varphi(v)) \in E_G$.

Generalizing adjacency preserving to reachability preserving and that again to $\sqcup$- and $\sqcap$-preserving results in the following notion of lattice homomorphism.

Definition 10. A lattice L_t is *homomorphic* to a lattice L_p if there is a function φ from L_p to L_t such that for $u, v \in L_p$, $\varphi(u \sqcap v)$ (similarly $\varphi(u \sqcup v)$) is reachable from $\varphi(u)$ and $\varphi(v)$ via pairwise node disjoint paths in L_t. If φ is injective we call it an *embedding* and we say L_t *embeds* L_p.

Definition 11. For a given lattice $L = (V, E)$ the *reachability graph* $R(L)$ is given by $R(L) = (V, \{(x, y) \mid x, y \in V$ and y is reachable from x in $E\})$.

$R(L)$ corresponds to the graph of the reflexive, transitive closure of the partial order of L. Transitive closure computation can be done via boolean matrix multiplication; the best known algorithm for computing the product of $n \times n$ bit-matrices works in $\mathcal{O}(n^{2.376})$ ([AHU74, HMR93]).

Theorem 12. *Let L_p, L_t be finite lattices. L_t embeds L_p iff $R(L_t)$ is (subgraph) homeomorphic to L_p.*

Proof. ($\Leftarrow$) Let φ be a homeomorphism from L_p to $R(L_t)$. If $(x, y) \in E_p$ then a simple path p in $R(L_t)$ exists, such that, $p = p_1 p_2 \ldots p_n$, $p_1 = \varphi(x)$ and $p_n = \varphi(y)$. Thus, $\varphi(y)$ is reachable from $\varphi(x)$ in $R(L_t)$ and therefore $\varphi(y)$ is reachable from $\varphi(x)$ in L_t, by Definition 11.

($\Rightarrow$) Let φ be a homomorphism from L_p to L_t. Define $E = \{(\varphi(x), \varphi(y)) \mid (x, y) \in E_p\}$. L_p is homeomorphic to $\varphi(L_p)$ because φ is an injection and every $(\varphi(x), \varphi(y))$ is a simple path. We have that $E \subseteq R(L_t)$, giving that $R(L_t)$ is homeomorphic to L_p. $\square$

Lemma 13. *For a lattice $L = (V, E)$ ($|E| > 0$) with $R(L) = (V_R, E_R)$, $|E_R| < |V|^2$.*

Proof. $\top_L$ can be reached from every node $v \in V$ resulting in $|V|$ edges in $R(L)$. In worst case, there may be at most one node $v_1 \in V \setminus \{\top_L\}$ reachable from at most $|V| - 1$ nodes (it cannot be reached from $\top_L$, otherwise there would be a cycle). By same argument there may be at most one node in $v_2 \in V \setminus \{\top_L, v_1\}$ reachable from at most $|V| - 2$ nodes etc. This results in $|E_R| \leq \sum_{i=1}^{|V|} i = \frac{|V|^2 + |V|}{2} < |V|^2$. $\square$

The above proof indicates that, in general, the transitive closure may introduce less edges in case of lattices than in case of graphs. We suspect that the upper bound $|V|^2$ could be lowered, but yet we do not have a proof.

Corollary 14. *Let L_p and L_t be a pattern and a test lattice, resp. If the homeomorphism problem for L_t and L_p is solvable in the order of complexity $\mathcal{O}(\rho(|V_t|, |E_t|))$ then the embedding problem is solvable in at most $\mathcal{O}(\rho(|V_t|, |E_t|^2))$, where ρ is a complexity function depending on L_t.*

As far as we know there is no problem in the graph theory equivalent to the one of lattice embedding. Probably one of the closely related problems is the subgraph isomorphism problem ([GJ79]). This is defined for two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ as the question whether there exists a subgraph of G isomorphic to H, that is, do subsets $V \subseteq V_G$ and $E \subseteq E_G$ exist such that $|V| = |V_H|$, $|E| = |E_H|$, for which there is an injection φ from V_H to V satisfying $(u, v) \in E_H$ iff $(\varphi(u), \varphi(v)) \in E$. This problem is known to be solvable in polynomial time for G a forest and H a tree, but remains NP-complete if G is an acyclic directed graph and H a directed tree. The lattice embedding problem distinguishes itself from the subgraph isomorphism problem on two lattices L_1, L_2 by demanding that L_2 is a sublattice of L_1.

Theorem 15. *Lattice embedding is NP-complete.*

Proof. The theorem is an immediate consequence of a result on the embeddability of lattices ([Fol80]) and the NP-completeness of subgraph isomorphism ([GJ79]).

The proof ([Fol80]) is based on the proposition that for any set C of positive integers there exists an infinite graph G, each vertex of which is incident with an infinite number of edges, such that C and $S(G)$ (the set of induced finite subgraphs of G, up to isomorphism) are polynomial time conjunctive equivalent. In [Fol80] an injective mapping is defined from a graph $G = (V, E)$ to a semilattice $D(G)$ on the carrier set $V \cup E \cup \{0\}$, where $0 \notin V \cup E$. The semilattice operation $\sqcap$ on distinct x, y is defined as follows: $x \sqcap y = z$ if $x, y \in V$, $z \in E$, and z has x, y as end vertices; $x \sqcap y = y$ if $x \in V$, $y \in E$, and x is incident with y; and $x \sqcap y = 0$ otherwise.

Clearly for finite H, $D(H)$ can be computed in polynomial time. It can be verified that $H \in S(G)$ iff $D(H)$ can be embedded in $D(G)$. (By adding a new $\top$ element to $D(G)$ and defining the binary operation $\sqcup$, we get an injection from a graph to a lattice.)

Finally, it has to be shown that the embedding of a finite (semi)lattice in $D(G)$ can be verified in polynomial time. This follows from a theorem of [Eva51] which proves that the word problem for the varieties of (semi)lattices is polynomial time solvable. □

4 An exponential algorithm for lattice embedding

A "brute-force" algorithm for lattice embedding would "blindly" compute all injections between the pattern and the test lattice satisfying the isomorphism condition. The algorithm given in this section is more efficient as it benefits from properties of the lattice structure. Therefore we expect that in practice this algorithm will perform better than a "brute-force" one.

Our algorithm, given in Fig. 3, uses a test lattice represented by a reachability closed edge set (cf. Sect. 3.3) and a pattern lattice. The node set of the pattern lattice is partitioned in sets of equal depth.

The partitioning algorithm is given in Fig. 2. It computes for each node the depths of all of its "parents" and assigns the maximum of these depths increased by 1 to the depth of that node. The set of parents of a node v is denoted by $out(v)$, where $out(v) = \{w \mid (v, w) \in E\}$ assuming that E is defined by covering pairs.

Input A lattice $L = (V, E)$ and $out(v)$, the set of nodes v is adjacent to, for $v \in V$.
Output $depth(v)$, for $v \in V$.

```
compute_depths:
begin
     depth(⊤) := 1;
     D := {⊤};
     while not D = V do
          for v ∈ V \ D do
               if out(v) ⊆ D then
                    D := D ∪ {v};
                    depth(v) = max{depth(v') | v' ∈ out(v)} + 1
               endif
          od
     od
end
```

Fig. 2. An algorithm for depth computation

Lemma 16. *Depth computation (cf.* Fig. 2*) is terminating, correct and has the complexity of* $\mathcal{O}(|V|^2)$.

Proof. (**termination**) The algorithm may not terminate only if in the while-loop D never equals V. Suppose this is the case. Then there is at least one node v for which $out(v) \not\subseteq D$. This means that there is at least one w for which $(v, w) \in E$ and $w \notin D$. Therefore $out(w) \not\subseteq D$ and there must be a $(w, z) \in E$ with $z \notin D$. By repeating this process we get that there exists a path $wz \ldots \top$, hence $\top \notin D$ which is certainly not true. Therefore every node $v \in V$ will eventually be included in D giving that the algorithm terminates.

(**correctness**) The proof is by induction on the structure of the lattice. Clearly, $depth(\top) = 1$. Now take a node v for which the depths of all of its parents have been computed. By definition, the depth of v is the length of a longest path from v to $\top$. If $vv_1v_2 \ldots \top$ is such a path then v_1 is a parent of v and $v_1v_2 \ldots \top$ is a longest path from v_1 to $\top$. By the induction hypothesis this length is $depth(v_1)$ and as this is the greatest length for all parents of v, $depth(v) = depth(v_1) + 1$.

(**complexity**) From the termination argument follows that the while-loop is executed maximal $|V|$ times. The for-statement has to consider a maximum of $|V|$ nodes for each node in $V \setminus D$. Hence, the if-clause is executed at most $|V|^2$ times. □

Let us return to the lattice embedding algorithm. The main procedure (*match*) starts the search at the top element of L_p with an empty matching set. The procedure consists of three parts. In lines 03-06 matchings are made for $C_1 = \{\top_p\}$ with all nodes of L_t. The second if-clause (lines 07-12) finds matches for all other classes of L_p. Here a matching for the classes $C_1, \ldots, C_k$ $(1 \leq k < n)$ that have been computed in a previous match is extended with a matching for C_{k+1}, if one exists. We have that for every element u from C_{k+1} there is at least one edge (u, v) in E_p with $v \in C_k$. The candidates for a match with C_{k+1} are therefore those nodes of V_t that are "below" the nodes of V_t used in the matchings for $C_1, \ldots, C_k$. By "below" we mean those elements of V_t that can reach an element from C_k and can not be reached from any element of V_t already used in the matching. In the partial order of L_t this corresponds to the union of the down-sets of the elements from V_t used in the match.

In line 08 elements of V_t already used in the matching are collected into M_k. The set of all "lower" elements of M_k are then assigned to $\bigtriangledown(M_k)$, in line 09. Now, all possible injections between C_k and $\bigtriangledown(M_k)$ have to be tried as candidates for a matching (lines 10 to 12 of the for-clause). A mapping m between C_k and $\bigtriangledown(M_k)$ is only allowed if it preserves reachability, that is, if a node $u \in C_{k+1}$ is adjacent to a node $v \in V_p$ then the matching for v must be reachable from $m(u)$. The adjacency condition is verified in the if-clause (line 11). If this condition is satisfied then we accept the matching for the classes $C_1, \ldots, C_k$ and extend it for the class C_{k+1} (line 12).

If all classes have been matched the computed match has to be verified for being a sublattice of L_t. This is done in lines 13-18 by a (polynomial) "brute-force" algorithm. All joins and meets of the match are computed and verified whether or not they are fulfilled in the matching. If so, we found a sublattice and the algorithm yields the matching M in the output.

Example 2. Consider the lattices L_t and L_p of Fig. 4. Partitioning of V_p by depth yields $\{(A), (B, C), (D)\}$. Below we show the stepwise computation of the match $M = \{(A, a), (B, c), (C, b), (D, e)\}$.

$(k = 1)$ $m = \{(A, a)\}$
$(k = 2)$ $M_2 = \{a\}$, $\bigtriangledown(M_2) = \{b, c, d, e, f, g\}$; $m = \{(B, c), (C, b)\}$
This is a well connected match because
$(C, A) \in E_p : (b, a) \in E_t^*$ and $(B, A) \in E_p : (c, a) \in E_t^*$
$(k = 3)$ $M_3 = \{a, b, c\}$, $\bigtriangledown(M_3) = \{d, e, f, g\}$; $m = \{(D, e)\}$
This is a well connected match because
$(D, B) \in E_p : (e, c) \in E_t^*$ and $(D, C) \in E_p : (e, b) \in E_t^*$
$(k = 4)$ $M_4 = \{a, b, c, e\}$

At this point we have that all elements of V_p have an image in L_t. The matching has to be verified for being a sublattice of L_t. As $d = c \sqcap b = b \sqcap c$ and $d \notin M_4$ we have to reject this matching for not being a sublattice. If however, we had (D, d) instead of (D, e) the matching would have been accepted. □

Input A test lattice $L_t = (V_t, E_t^*)$ with E_t^* the reachability closure of E_t (cf. Definition 11) and a pattern lattice $L_p = (V_p', E_p)$ with V_p' partitioned in classes $C_1, .., C_n$ where $C_i = \{v \in V_p \mid depth(v) = i\}$, for $1 \leq i \leq n$.

Output All sublattices of L_t isomorphic to L_p represented by sets of tuples M, where a tuple $(x, y) \in M$ denotes the matching of $x \in L_p$ with $y \in L_t$.

```
01 procedure match (k, M):
02 begin
03     if $k = 1$ then
04         for $v_t \in V_t$ do
05             $m := \{(\top_p, v_t)\}$;
06             match($k+1$, $M \cup m$) od
07     elseif $1 < k \leq n$ then
08         $M_k := \{b \mid (a,b) \in M\}$;
09         $\bigtriangledown(M_k) := \{a \mid \exists_{c \in M_k} : (a,c) \in E_t^*$ and $a \notin M_k\}$;
10         for every matching $m$ between $C_k$ and a subset $X$ of $\bigtriangledown(M_k)$
                   with $|X| = |C_k|$ do
11             if $\forall_{a \in C_k} \forall_{b \in out(a)}$
                   $[(a,x) \in m$ and $(b,y) \in M \Rightarrow (x,y) \in E_t^*]$ then
12                 match($k+1$, $M \cup m$)
               endif od
13     else
           { $k$ corresponds to $n+1$ }
14         lattice condition := true;
15         for $(a_1, b_1) \in M$, $(a_2, b_2) \in M$ do
16             if $\neg\exists a$: $(a, b_1 \sqcup b_2) \in M$ or $(a, b_1 \sqcap b_2) \in M$ then
17                 lattice condition := false endif od
18         if lattice condition then output($M$) endif
19     endif
20 end

21 begin
22     match(1, $\emptyset$)
23 end
```

Fig. 3. A top down algorithm for lattice inclusion

About termination, correctness and complexity of our lattice embedding algorithm the following can be said:

(**termination**) The set of permutations between L_p and L_t is finite. As the algorithm computes a subset of this set and no permutation is computed more than once the algorithm must terminate.

(**correctness**) A matching computed by the algorithm is a bijection between V_p and a subset of V_t. The sublattice conditions are trivially verified by lines 14-17.

The other way round, it can easily be seen that the algorithm finds every sublattice of L_t. Take, for instance, a sublattice of L_t having $v \in V_t$ as top element. At some point the top of L_p will be matched with v (lines 04-06). Let Y be the part of the sublattice that corresponds to the isomorphic image of C_2 from L_p. Clearly, Y is "below" v,

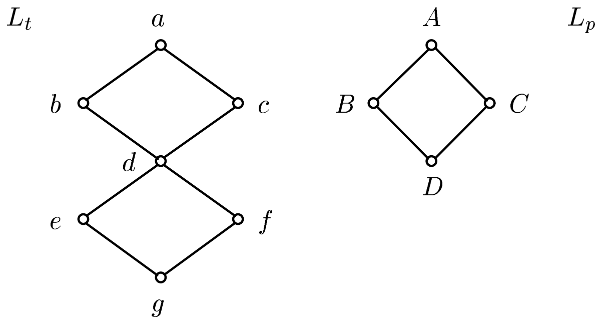

Fig. 4. Sample test and pattern lattices

$Y \subseteq \bigtriangledown(M_2)$ and so, C_2 will be matched with Y in line 10. The above argument holds also for all other classes $C_3, \ldots, C_n$, therefore the algorithm finds the sublattice (if there is any).

(**complexity**) We may take $|V_t|!/(|V_t| - |V_p|)!$ as upper bound for the amount of matchings to be computed (which equals the number of injections from V_p to V_t). The computation of the sets and conditions are all at most of a squared polynomial complexity in L_t. M_k (in line 08) can be constructed in one pass through M. Since $|M| \leq |V_p|$, we have that M_k is $\mathcal{O}(|V_p|)$. The condition in the if-clause (line 11) can be checked by examining all edges of E_p for every node of C_k. This can be done in $|E_p| * |C_k| \leq |E_p|^2$ steps, which is in the order of $\mathcal{O}(|E_p|^2)$.

Finally, the complexity of the lattice conditions for a lattice of size n has to be computed. Joins and meets can be computed in $\mathcal{O}(n)$ ([Mar80]). The only known case for which $\mathcal{O}(\log n)$ complexity can be achieved are the boolean lattices ([Hab96]). In our algorithm joins and meets are computed for all pairs of L_p resulting in the complexity of $\mathcal{O}(|V_p|^2|V_t|)$.

4.1 Lowering the complexity of lattice embedding

In this section we reduce the complexity of our algorithm to $\mathcal{O}((t!/(t-i_p)!)p^2t)$, where t is the size of the test lattice, p the size of the pattern lattice size and i_p denotes either of the number of $\sqcup$- or $\sqcap$-irreducible elements of the pattern lattice. This is due to the fact that a lattice is determined by its irreducible elements [Dil82, Bir67]. Therefore an injection from irreducible elements of the pattern lattice to elements of the test lattice is sufficient for determining a matching. We show this for the $\sqcup$-irreducibility case. For $\sqcap$-irreducibility similar arguments apply.

Lemma 17. ([Wil77]) *Every element in a (finite) lattice has an irredundant representation as a join (meet) of $\sqcup$-irreducible ($\sqcap$-irreducible) elements.*

This means that every element in a lattice can be considered as a meet- or join-junction of $\sqcap$-irreducible and $\sqcup$-irreducible elements, resp.

Irreducible elements of a lattice can be determined easily from its Hasse diagram as stated by Lemma 18 and 19.

Lemma 18. *Let L be a lattice and x an element of L. Then x is $\sqcup$-irreducible iff x has outdegree 1; and x is $\sqcap$-irreducible iff x has indegree 1.*

Lemma 19. *Computing the sets of $\sqcup$- and $\sqcap$-irreducible elements of a lattice of size n can be done in $\mathcal{O}(n)$ steps.*

The following lemma states that any element of a lattice can be given as a $\sqcup$-junction of $\sqcup$-irreducible elements appearing in the element's down-set.

Lemma 20. *Let L be a lattice and $x \in L$. Define the set $J(x)$ as $J(x) = \{y \in L|\ y$ is $\sqcup$-irreducible and $y \leq x\}$. Then $x = \sqcup J(x)$.*

Proof. Let $x \in L$. If x is $\sqcup$-irreducible then x is the least upper bound of $J(x)$ since for all $y \in J(x), y \leq x$ and $x \in J(x)$.

Otherwise, x is not $\sqcup$-irreducible. Then $x = y_1 \sqcup y_2$ for some $y_1, y_2 \in L$. If both y_1 and y_2 are $\sqcup$-irreducible then we have finished. Otherwise, we go on recursively with y_1 or y_2 (or both) until only $\sqcup$-irreducible elements remain. Since all these $\sqcup$-irreducible elements are contained in $J(x)$, we conclude that $x = \sqcup J(x)$. □

Lattice matching with the $\sqcup$-irreducible elements of the pattern lattice determines the matching with the whole pattern lattice.

Lemma 21. *Let m be a matching from a pattern lattice L_p to a test lattice L_t such that L_p is isomorphic to $m(L_p)$ and $m(L_p)$ is a sublattice of L_t. Then for all $x \in V_p$ and for $J(x)$ holds: $m(x) = \sqcup m(J(x))$.*

Proof. By Lemma 20 we can represent x by $x = x_1 \sqcup x_2 \sqcup \ldots \sqcup x_q$ for $q \leq |J(x)|$ and $x_i \in J(x)$, $1 \leq i \leq q$. Let $y \in V_t$ be defined by $y = m(x_1) \sqcup m(x_2) \sqcup \ldots \sqcup m(x_q)$.

By the sublattice condition we have that there must be a $z \in V_p$, such that, $m(z) = y$. By isomorphism of m follows that $z = x$. □

Let i_p denote the amount of $\sqcup$-irreducible elements of L_p. By Lemma 21 we only have to consider $t!/(t - i_p)!$ injections from the $\sqcup$-irreducible elements of L_p to L_t. Computing the matches for the remaining elements of L_p costs for each element at most $\mathcal{O}(i_p\, t)$, since we have to consider a $\sqcup$-junction for at most i_p number of $\sqcup$-irreducible elements and the computation of a join costs $\mathcal{O}(t)$. There are $p - i_p - 2$ of such elements (the '2' comes from the top and bottom of L_p) giving a total of $\mathcal{O}((p - i_p - 2)t)$. With respect to the sublattice conditions we only need to verify the meets of the matching since the join conditions are valid by construction of the matchings. The costs for the meet-conditions remain $p^2 t$. Therefore the total costs are (with constant c):

$$
\begin{aligned}
&\leq\ c\frac{t!}{(t - i_p)!}[(p - i_p - 2)i_p\, t) + p^2 t] \\
&\leq\ c\frac{t!}{(t - i_p)!}[p^2 t + p^2 t]
\end{aligned}
$$

which is $\mathcal{O}(\frac{t!}{(t - i_p)!}p^2 t)$, not considering constants.

5 Application

Our research for lattice embedding was motivated by results in concept (or Galois) lattices ([Wil82]). The close relationship between (a more general class of) concept lattices and conceptual graphs is clarified in [LW95].

A *concept lattice* is a representation of a (e.g. binary) relation R between a set of objects G and a set of attributes M. The triple (G, M, R) is called the *context*. The lattice arises from that context by applying a Galois connection between the power sets of G and M. This Galois connection is a particular one, called a *polar* ([Bir67]).

Various applications of concept lattices have been reported in the literature ([MG92, GMA91, Wil89]). Typically, such an application assumes a subset of G (resp. M) to be given as *input* for which a corresponding subset of M (resp. G) is computed (if such exists) by searching the concept lattice (*knowledge base*) for a concept having the smallest extent containing the given subset of G. So, for e.g. given input $\mathsf{A}' \subseteq G$ the answer is a concept (A, B) of the concept lattice, such that $\mathsf{A}' \subseteq \mathsf{A}$ and A is "smallest". Then B is the corresponding subset of M. In this sense a concept lattice is an appropriate representation for what is referred to in AI as "cooperative communication"([Gri75]).

Motivated by different practical problems of e.g. processing of visual input, recognition of continuous speech ([Far94]) and knowledge representation, an extension to the above model was introduced in [SF94]. The essence of this extension is that a set of interrelated subsets of objects *and* attributes are allowed to be given as input. It turns out that in this case the input itself can be represented by means of a Galois connection, as well. Eventually this generalization allows the input to be a concept lattice.

For such an input, similarly to our previous example, the answer is the "smallest" sublattice of the concept lattice (knowledge base) that subsumes it. In sum, we have that an embedding for two concept lattices has to be determined (the task of finding a "smallest" such lattice can be handled separately).

The embedding problem for concept lattices is not only a topological problem. The semantics of the nodes of the lattices must be taken into consideration as well. We say a concept lattice (L_t) embeds another concept lattice (L_p), if (lattice) L_t embeds (lattice) L_p and an injective mapping, from the context of L_p to the context of L_t (restricted to that embedding), exists. Due to space limitations we can not elaborate the definition further; a formal treatment of this can be found in [SF95].

In general, we may have any number of such L_p concept lattices. In [SF95] we showed that in that case and under certain conditions the embedding problem "becomes" a decomposition problem for concept lattices which involves the recognition of the given L_p concept lattice(s) as *sublattice(s)* of the given input concept lattice L_t. Whenever such a decomposition is possible we may get an abstract conceptualization of the input.

Interestingly, the concept lattice construction can be repeatedly applied, now to the structure resulted by such a decomposition. In [SF95] we showed that, due to simple properties of a Galois connection, various ways for a recursive application are possible.

Let us make our last statement more precise. Assume that having applied a Galois connection we constructed a concept lattice. A sufficient condition for the (recursive) application of a Galois connection is that a set of objects and attributes and a relation between them are given, for now, in the concept lattice. Recall that these three components are parameters of the Galois connection, and therefore, can be arbitrarily chosen. It turned out that taking the sublattices of a concept lattice as objects and their related sublattices as attributes can be most practical. Informally, two sublattices are *related* if there exists a concept (not containing the meaningful empty set) shared by both sublattices.

5.1 Example

In what follows we assume some familiarity with Galois lattices. We illustrate the construction of a Galois lattice with two examples taken from plain geometry ([SF95]). These examples, which are specifications of geometrical notions represented as Galois lattices, make use of two *primitive* notions: the point, and the alignment of a line on point(s).

Our first example is the angle. We specify an angle as a collection of three points and two lines lying on them as shown in figure 5. We choose as objects, the points P_0, P_1 and P_2, and as attributes, the alignment on straight lines e and f. Alignment of a point on a line x is denoted by on_x. The members of the Galois lattice (L_{angle}) are:

$$C_0 = (\{P_0\}, \{on_e, on_f\}) \ , C_1 = (\{P_0, P_1, P_2\}, \{\ \}) \ ,$$
$$C_2 = (\{P_0, P_1\}, \{on_e\}) \ , \ \ C_3 = (\{P_0, P_2\}, \{on_f\}) \ .$$

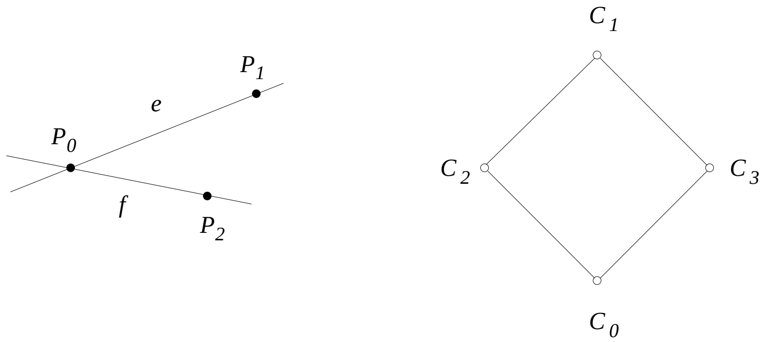

Fig. 5. Diagram and Galois lattice of an angle

The second example is the quadrangle. Following the same procedure as above, we get that the objects are the points P_0–P_3, and the attributes the alignment on straight lines e, f, g and h. The diagram of the quadrangle and its Galois lattice (L_{quadr}) are given in figure 6. The Galois lattice has the following elements:

$$\begin{array}{ll}
C_0 = (\{\ \}, \{on_e, on_f, on_g, on_h\}) \ , & C_1 = (\{P_0, P_1, P_2, P_3\}, \{\ \}) \ , \\
C_2 = (\{P_0\}, \{on_e, on_f\}) \ , & C_3 = (\{P_1\}, \{on_f, on_g\}) \ , \\
C_4 = (\{P_2\}, \{on_g, on_h\}) \ , & C_5 = (\{P_3\}, \{on_e, on_h\}) \ , \\
C_6 = (\{P_0, P_1\}, \{on_f\}) \ , & C_7 = (\{P_0, P_3\}, \{on_e\}) \ , \\
C_8 = (\{P_1, P_2\}, \{on_g\}) \ , & C_9 = (\{P_2, P_3\}, \{on_h\}) \ .
\end{array}$$

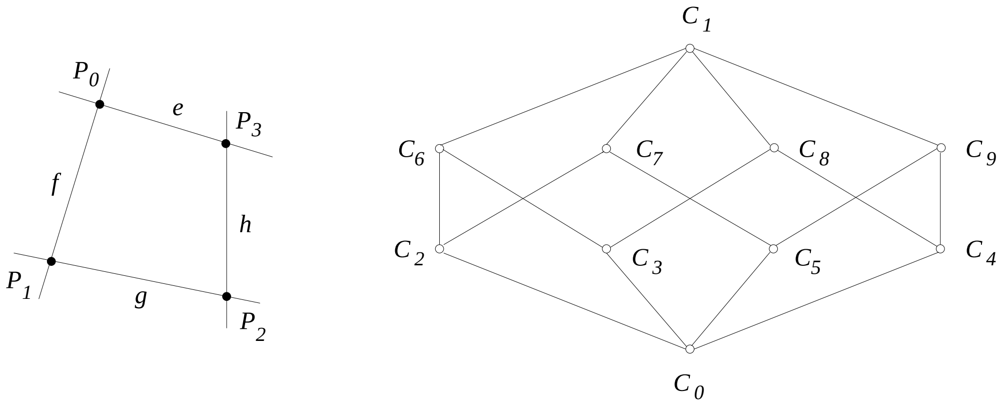

Fig. 6. Diagram and Galois lattice of a quadrangle

Now, if we wish to find out whether the quadrangle has angles, then we must realize that, though all necessary information is present in the specifications they are not immediately applicable. A simple-minded solution would, for instance, extend the specification of the quadrangle and add the conditions for points and lines (of the quadrangle) to form an angle. By doing this we add, in fact, a copy of the specification of the angle to the one of the quadrangle.

In the remaining of this section we show that the use of concept lattices allows a more elegant solution to this problem. Clearly, this problem is just an example; the method we propose can be used for a wide range of problems where some structure is to be "found" as part of another one.

Let us apply our algorithm from figure 3 to L_{quadr} and L_{angle}. This yields that there are eight lattice embeddings, but only four of them correspond to occurrences of the Galois lattice of the angle (in L_{quadr}). These are the sublattices $A_0 = \{C_1, C_2, C_6, C_7\}$, $A_1 = \{C_1, C_3, C_6, C_8\}$, $A_2 = \{C_1, C_4, C_8, C_9\}$ and $A_3 = \{C_1, C_5, C_7, C_9\}$. Let us recall that the specification of the quadrangle contains *no* mentioning of its angles. Nevertheless we were able to find them, due to our representation and algorithm.

We have seen that the lattice of the quadrangle contains the sublattices A_0–A_3. Let us take them now as our objects and as attributes the sublattices related to them. Eventually we get the relation: $\{(A_0, A_1)$, (A_0, A_3), (A_1, A_0), (A_1, A_2), (A_2, A_1), (A_2, A_3), (A_3, A_0), $(A_3, A_2)\}$. By this we completely defined a new context.

By applying the Galois connection again, to the new context, we get the concepts:

$$\begin{aligned} C_0 &= (\{A_0, A_1, A_2, A_3\}, \emptyset) \ , \ C_1 = (\{A_0, A_2\}, \{A_1, A_3\}) \ , \\ C_2 &= (\{A_1, A_3\}, \{A_0, A_2\}) \ , \ C_3 = (\emptyset, \{A_0, A_1, A_2, A_3\}) \ . \end{aligned}$$

This lattice is isomorphic to the one of the angle (cf. figure 5), as we can show it by applying our lattice embedding algorithm, but it has now a completely different meaning: a quadrangle conceptualized as pairs of angles. Indeed, we observe that C_1 and C_2 (those concepts that do not contain the meaningless empty set) have a pair of angles as object resp. attribute sets.

Summary and conclusions

This paper describes lattices embedding, which is NP-complete, and exemplifies its use. We start with a naive matching algorithm and define an efficient one. The complexity of lattice embedding is exponential, however, various applications of Galois lattices ([MG92, GM93]) indicate that the practical size of such lattices do not grow exponentially with the size of their context. This implies that our algorithm can be practical, as well. Finally, the examples in section 5 give evidence that the Galois lattice representation together with lattice embedding as a primary operation may allow a uniform and systematic approach to knowledge representation.

Acknowledgements

We acknowledge valuable correspondence with K. Regan and L. Nourine. We are indebted to M. Habib and the referees for their very useful comments and suggestions.

References

[AHU74] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.

[Bir67] G. Birkhoff. *Lattice theory.* American Mathematical Society, 1967.

[CK92] F. Costello and M.T. Keane. A model-based theory of conceptual combination. In K. Ryan and R. Sutcliffe, editors, *AI and Cognitive Science.* Springer-Verlag, 1992.

[Dil82] R.P. Dilworth. The role of order in lattice theory. In I. Rival, editor, *Ordered sets*, pages 333–353. Dordrecht-Boston, 1982.

[DP90] B.A. Davey and H.A. Priestley. *Introduction to lattices and order.* Cambridge University Press, 1990.

[Eva51] T. Evans. The word problem for abstract algebras. *J. of London Math Soc.*, 26:64–71, 1951.

[Far94] J. Farkas. *(personal communication)*, 1994.

[FKMP95] M.R. Fellows, J. Kratochvil, M. Middendorf, and F. Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13(3):266–282, 1995.

[Fol80] S. Foldes. On the complexity of the lattice embeddability problem. *Utilitas Mathematica*, 17:79–84, 1980.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractability, A guide to the theory of NP-completeness.* W.H. Freeman and co., 1979.

[GM93] R. Godin and H. Mili. Building and maintaining analysis-level class hierarchies using Galois lattices. *ACM SIGPLAN Notices*, pages 394–410, 1993.

[GMA91] R. Godin, R. Missaoui, and H. Alaoui. Learning algorithms using a Galois lattice structure. In *Proc. of the 1991 IEEE Int. Conf. on tools of AI*, pages 22–29, San Jose (CA), 1991.

[Gri75] H.P. Grice. Logic and conversation. In P. Cole and J. Morgan, editors, *Syntax and semiotics: Speech acts*, volume 3, pages 41–58, New York, 1975. Academic Press.

[Hab96] M. Habib. *(personal communication)*, 1996.

[HMR93] M. Habib, M. Morvan, and J.X. Rampon. On the calculation of transitive reduction-closure of orders. *Discrete Mathematics*, 111:289–303, 1993.

[Joh82] D.S. Johnson. The np-completeness column: An ongoing guide. *J. of Algorithms*, 3(3):89–99, 1982.

[LW95] F. Lehmann and R. Wille. A triadic approach to formal concept analysis. In G. Ellis, R. Levinson, W. Rich, and J.F. Sowa, editors, *Third Int. Conf. on Conceptual Structures, ICCS'95*. Springer-Verlag, 1995.

[Mar80] G. Markowsky. The representation of posets and lattices by sets. *Algebra Universalis*, 11:173–192, 1980.

[MG92] R. Missaoui and R.G. Godin. An incremental concept formation approach for learning from databases. In *Proc. of the Workshop on Formal Methods of Databases and Software Engineering*, pages 39–53, Montreal, 1992.

[SF94] J.J. Sarbo and J.I. Farkas. Concept sublattices. In *Proc. of the European Conference on Machine Learning (ECML'94)*, Catania, Italy, 1994.

[SF95] J.J. Sarbo and J.I. Farkas. Knowledge representation and acquisition by concept lattices. In Shaul Markovitch, editor, *Proc. of the 11th Izraeli Symposium on Artificial Intelligence (ISAI'95)*, Hebrew University of Jerusalem, Izrael, 1995.

[Wil77] R. Wille. Aspects of finite lattices. In M. Aigner, editor, *Higher combinatorics*, pages 79–100. Reidel Publ. Company, 1977.

[Wil82] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. D. Reidel, Dordrecht-Boston, 1982.

[Wil89] R. Wille. Knowledge acquisition by methods of formal concept analysis. In E. Diday, editor, *Proc. of the Conf. on Data Analysis, Learning Symbolic and Numeric Knowledge*, pages 365–380. Nova Science Publishers, Inc., 1989.