Hermes: an efficient algorithm for building Galois sub-hierarchies

Anne Berry, Marianne Huchard, Amedeo Napoli, Alain Sigayret

Hermes: an efficient algorithm for building Galois sub-hierarchies

Anne Berry[1], Marianne Huchard[2], Amedeo Napoli[3], and Alain Sigayret[1]

[1] LIMOS - CNRS UMR 6158 - Université Clermont-Ferrand II (France)**
{berry, sigayret}@isima.fr
[2] LIRMM - CNRS UMR 5506 - Université de Montpellier II - Montpellier (France)
huchard@lirmm.fr
[3] LORIA - CNRS UMR7503 - Vandoeuvre-lès-Nancy (France)
amedeo.napoli@loria.fr

Abstract. Given a relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ on a set $\mathcal{O}$ of objects and a set $\mathcal{A}$ of attributes, the Galois sub-hierarchy (also called AOC-poset) is the partial order on the introducers of objects and attributes in the corresponding concept lattice. We present a new efficient algorithm for building a Galois sub-hierarchy which runs in $O(min\{nm, n^\alpha\})$, where n is the number of objects or attributes, m is the size of the relation, and n^α is the time required to perform matrix multiplication (currently $\alpha = 2.376$).

1 Introduction

Galois lattices (also called concept lattices) are a powerful tool for data modeling. Such a lattice is built on a relation between a set of objects and a set of attributes. The main drawback of this structure is that it may have an exponential size in the number n of objects or attributes. A canonical sub-order of the lattice, its Galois sub-hierarchy (GSH, also called AOC-Poset), of much smaller size, is recommended whenever possible. This GSH preserves only the key elements of the lattice: object-concepts and attribute-concepts (also called introducers); the number of these key elements is at most equal to the total number n of objects and attributes.

Galois sub-hierarchies were introduced in software engineering by Godin et al. [10] for class hierarchy reconstruction and successfully applied in later research work (see e.g. [15]). The AOC-poset (Attribute/Object Concepts poset [9]) has been used in applications of FCA to non-monotonic reasoning and domain theory [12], and to produce classifications from linguistic data [18, 20]. Specific parts of the GSH (mainly attribute-concepts) have been used in several works, including approaches for refactoring a class hierarchy [14] and recently for extracting a feature tree from a set of products in Software Product Lines [21].

** Research partially supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010.

© Laszlo Szathmary, Uta Priss (Eds.): CLA 2012, pp. 21–32, 2012.
ISBN 978–84–695–5252–0, Universidad de Málaga (Dept. Matemática Aplicada)

Three algorithms for building GSH already exist: Ares [7], Ceres [14], and Pluton [1]. Each of them has a time complexity of $O(n^3)$, and they are somewhat complicated. Their comparative experimental running times were investigated in [1].

In this paper, we present a new algorithm for building Galois sub-hierarchies, which we call Hermes, with a better complexity. Hermes runs in $O(nm)$ time, where m is the size of the relation, and is very easy to understand and implement. With more effort invested in the implementation, Hermes can be made to run in $O(n^\alpha)$ (*i.e.* $O(n^{2.376})$) time, which is the time for performing matrix multiplication. Hermes works by simplifying and then extending the input relation into a relation which contains in a compact fashion all the necessary information on the elements of the GSH.

The paper is organized as follows: after this introduction, we give some notations and definitions. Section 3 briefly outlines how previous algorithms work. Section 4 proves some preliminary results and presents the algorithmic tools necessary to ensure our good complexity. Section 5 describes and analyzes in detail the successive steps of our algorithmic process. Section 6 gives the algorithm. Section 7 describes the special case for chordal bipartite relations, where the final relation can easily be obtained in $O(n^2)$ time. We conclude in Section 8.

2 Definitions and notations

The different communities handling Galois lattices and Galois sub-hierarchies use various notations. Here, we will use algebraic notations detailed below.

Given two finite sets $\mathcal{O}$ (of 'objects') and $\mathcal{A}$ (of 'attributes'), a binary **relation** $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ indicates which objects of $\mathcal{O}$ are associated with which attributes of $\mathcal{A}$. $\mathcal{O}$ is called the **starting set** of the relation. We will denote $n = |\mathcal{O}| + |\mathcal{A}|$ and $m = |\mathcal{R}|$. For $(x, y) \in \mathcal{R}$, we will say that x is an **antecedent** of y, and y is an **image** of x. For $x \in \mathcal{O}$, $\mathcal{R}(x) = \{y \in \mathcal{A} \,|\, (x, y) \in \mathcal{R}\}$ is the **image set** (**row**) of x, and for $y \in \mathcal{A}$, $\mathcal{R}^{-1}(y) = \{x \in \mathcal{O} \,|\, (x, y) \in \mathcal{R}\}$ is the **antecedent set** (**column**) of y. Note that notation x' is used in FCA [9] for $R(x)$ and $R^{-1}(x)$. The term **line** will be indifferently used for row and column. The triple $(\mathcal{O}, \mathcal{A}, \mathcal{R})$ is called a **context**.

A maximal rectangle of $\mathcal{R}$, also called a **concept**, is a maximal Cartesian sub-product of $\mathcal{R}$, *i.e.* $X \times Y$ such that $\forall x \in X$, $\forall y \in Y$, $(x, y) \in \mathcal{R}$ and $\forall w \in \mathcal{O} - X$, $\exists y \in Y \mid (w, y) \notin \mathcal{R}$ and $\forall z \in \mathcal{A} - Y$, $\exists x \in X \mid (x, z) \notin \mathcal{R}$. X is called the **extent** and Y the **intent** of concept $X \times Y$, which is denoted (X, Y). In our examples, we may omit set brackets when the meaning is clear. The extent and intent of concept C will be denoted **Extent(**C**)** and **Intent(**C**)**. The concepts, ordered by inclusion on their extents (or dually by inclusion on their intents) form a lattice $\mathcal{L}(\mathcal{R})$ called a **Galois lattice** or a **concept lattice**. For two concepts C and C', $C <_{\mathcal{L}(\mathcal{R})} C'$ will denote Extent(C)$\subset$Extent(C'). A lattice is represented by its **Hasse diagram**, where reflexivity and transitivity edges are omitted.

An **object-concept** is a concept C_x which *introduces* some object x: x is in the extent of C_x but is not in the extent of any smaller concept $C' <_{\mathcal{L}(\mathcal{R})} C_x$. Dually, an **attribute-concept** is a concept C_y which *introduces* some attribute y: y is in the intent of C_y but is not in the intent of any greater concept $C' >_{\mathcal{L}(\mathcal{R})} C_y$. Thus, the intent of object-concept C_x is $\mathcal{R}(x)$, and the extent of attribute-concept C_y is $\mathcal{R}^{-1}(y)$. Object-concepts and attribute-concepts are also called **introducer concepts** or simply **introducers**. Objects are introduced from bottom to top and attributes from top to bottom in $\mathcal{L}(\mathcal{R})$. A given concept may introduce several objects and/or attributes. Note that [9] uses arrow relations to characterize the relationship between attribute-concepts and object-concepts, but without referring to Galois sub-hierarchies.

A relation is said to be **clarified** when it has no identical lines. A relation is said to be **reduced** when it is clarified and has no line which is the intersection of several other lines. When a relation is reduced, the irreducible elements of the lattice are exactly the introducers, whereas in a non-reduced relation there will be extra introducers.

$\mathcal{H}(\mathcal{R})$ denotes the **Galois sub-hierarchy** (GSH) of relation $\mathcal{R}$, defined by the set of introducer concepts ordered as in $\mathcal{L}(\mathcal{R})$. $\mathcal{H}(\mathcal{R})$ is then a sub-order of $\mathcal{L}(\mathcal{R})$. The elements of $\mathcal{H}(\mathcal{R})$ are generally labeled by the objects and/or attributes they introduce, defining the **simplified labeling**. The same simplified labeling applies to $\mathcal{L}(\mathcal{R})$, in which some concepts may have an empty label. $<_{\mathcal{H}(\mathcal{R})}$ will be used to compare two elements of $\mathcal{H}(\mathcal{R})$, as $<_{\mathcal{L}(\mathcal{R})}$ is used for $\mathcal{L}(\mathcal{R})$.

A **linear extension** of a partially ordered set P is a total order in which P is included.

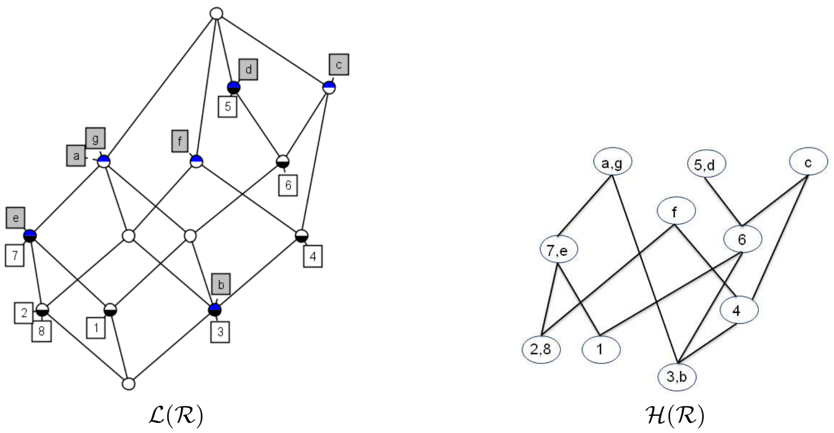
Running example. Figure 1 shows the Galois lattice $\mathcal{L}(\mathcal{R})$ (as drawn by Context Explorer [24]) and the Galois sub-hierarchy $\mathcal{H}(\mathcal{R})$ of relation $\mathcal{R}$. In $\mathcal{L}(\mathcal{R})$, concept $(1, acdeg)$ introduces 1 (simplified label: 1), concept $(1346, c)$ introduces c (simplified label: c), and concept $(3, abcdfg)$ introduces 3 and b (simplified label: $3, b$). All these concepts are in $\mathcal{H}(\mathcal{R})$; concept $(13, acdg)$ introduces nothing (simplified label empty) and as such is not in $\mathcal{H}(\mathcal{R})$.

$\mathcal{R}$	a	b	c	d	e	f	g
1	×		×	×	×		×
2	×				×	×	×
3	×	×	×	×		×	×
4			×			×	
5				×			
6			×	×			
7	×				×		×
8	×				×	×	×

3 Previous algorithms

We present a brief description of the existing algorithms for building Galois sub-hierarchies; all run in $O(n^3)$ time, where n stands for the number of objects and attributes in the input relation. The reader is referred to the corresponding publications for detailed descriptions and to [1] for a comparative experimental study of those algorithms.

Pruned lattice. [10]
Pruned lattice is the name given to a Galois sub-hierarchy by [10] which is, to our best knowledge, the first paper defining this structure. [10] considers a specific



$\mathcal{L}(\mathcal{R})$ $\mathcal{H}(\mathcal{R})$

Fig. 1. Lattice $\mathcal{L}(\mathcal{R})$ and Galois sub-hierarchy $\mathcal{H}(\mathcal{R})$, both with the simplified labeling, for our running example.

case where each object owns a specific attribute (not owned by the others). A class inheritance hierarchy is flattened in a table that associates the classes with their members (class attributes and methods). The hierarchy is then rebuilt in a better factorized way, eliminating redundancy.

Pluton. [1]
This algorithm is composed of three successive processes: TomThumb, ToLinext, and ToGSH. TomThumb [3] produces an ordered list of the simplified labels of extents and intents, which maps to a linear extension of the GSH. ToLinext then searches this list to merge consecutive pairs consisting of a simplified extent and a simplified intent belonging to the same concept. Finally, ToGSH computes the edges of the Hasse diagram of the GSH.

Ceres. [14]
This algorithm computes at the same time the elements of the GSH and its Hasse diagram. The elements are computed in an order which maps to a linear extension of the Galois sub-hierarchy. In a first stage, the columns of the relation are sorted by decreasing number of crosses to generate the introducers by decreasing extent size. In the second stage, the strategy is twofold: compute the attribute-concepts by groups sharing the same extent, and add object-concepts when their intent is covered by the intents of the attribute-concepts already computed. The edges of the hierarchy are determined on-the-fly.

Ares. [7]
This algorithm is incremental: given a Galois sub-hierarchy and a new object with its attribute set S, the hierarchy is modified to include this new object. For this, the initial GSH is traversed using a linear extension. If I denotes the intent of the current (explored) concept, then four main cases may occur and the GSH will be updated accordingly: $I = S$, $I \subset S$, $I \supset S$, or I and S are not

comparable by set inclusion. If during exploration, the algorithm did not find an initial concept whose intent is S, a new concept is created. For every modification of the Hasse diagram, the algorithm removes newly created transitivity edges. At the same time, for each modified intent, the algorithm checks if some concept has a simplified label which is empty and removes such concept.

4 Preliminary results and algorithmic tools

4.1 Some preliminary results

The following theorem will help order the introducers:

Theorem 1.
Let C_x be the introducer of $x \in \mathcal{O}$ and C_y be the introducer of $y \in \mathcal{A}$.
– *$C_x \leq_{\mathcal{H}(\mathcal{R})} C_y$ iff $(x, y) \in \mathcal{R}$.*
– *$C_x \geq_{\mathcal{H}(\mathcal{R})} C_y$ iff Intent(C_x)$\subseteq$Intent(C_y) iff Extent(C_x)$\supseteq$Extent(C_y).*
In this case, $(x, y) \notin \mathcal{R}$ except if $C_x = C_y$.
– *Otherwise, C_x and C_y are not comparable.*

[22] introduced the notion of **domination** which originates from graph theory. Domination in a relation stemmed from the concept of domination in the co-bipartite graph which is the complement of the bipartite graph induced by the relation.

An attribute $y \in \mathcal{A}$ is said to **dominate** an attribute $z \in \mathcal{A}$ in $\mathcal{R}$ if the antecedent set of y is included in the antecedent set of z: $\mathcal{R}^{-1}(y) \subseteq \mathcal{R}^{-1}(z)$; the corresponding relation is denoted $Dom_{\mathcal{A}}$. When the inclusion is strict, the domination is said to be strict. For $y \in \mathcal{A}$, $Dom_{\mathcal{A}}(y) = \{z \in \mathcal{A} \,|\, \mathcal{R}^{-1}(y) \subseteq \mathcal{R}^{-1}(z)\}$. This preorder defines the way attributes label the concepts of $\mathcal{H}(\mathcal{R})$ from top (the dominat**ing** attributes) to bottom (the dominat**ed** attributes).

A domination relation, $Dom_{\mathcal{O}}$, can also be defined between objects by inclusion of their image sets: $\forall w \in \mathcal{O}$, $Dom_{\mathcal{O}}(w) = \{x \in \mathcal{O} \,|\, \mathcal{R}(w) \subseteq \mathcal{R}(x)\}$. The label of $\mathcal{H}(\mathcal{R})$ will be set from bottom (the dominating objects) to top (the dominated objects), according to the dual behavior of objects and attributes in concepts.

Theorem 2. *[4]*
Endowed with domination relation $Dom_{\mathcal{A}}$, the set of attribute-concepts of $\mathcal{R}$ forms a sub-order of $\mathcal{H}(\mathcal{R})$: for $y, z \in \mathcal{A}$, the introducer of y is smaller than or equal to the introducer of z iff $(y, z) \in Dom_{\mathcal{A}}$.
Endowed with domination relation $Dom_{\mathcal{O}}$, the set of object-concepts of $\mathcal{R}$ forms a sub-order of $\mathcal{H}(\mathcal{R})$ and $\mathcal{L}(\mathcal{R})$: for $w, x \in \mathcal{O}$, the introducer of w is greater than or equal to the introducer of x iff $(w, x) \in Dom_{\mathcal{O}}$.

Example. In our running example, $R^{-1}(b) = \{3\} \subset R^{-1}(a) = \{1, 2, 3, 7, 8\}$; attribute b dominates attribute a and the introducer of b is smaller than the introducer of a, as shown in Figure 1. $Dom_{\mathcal{A}}(b) = \{a, b, c, d, f, g\}$. $R(6) = \{c, d\} \subset R(1) = \{a, c, d, e, g\}$; object 6 dominates object 1 and the introducer of 6 is greater than the introducer of 1. $Dom_{\mathcal{O}}(6) = \{1, 3\}$.

4.2 Algorithmic tools

We will need two processes for our complexity results.

The first process is to rapidly recognize lines of a relation which are equal, which corresponds to the clarification of context $(\mathcal{O}, \mathcal{A}, \mathcal{R})$. This can be done in linear time $O(|\mathcal{R}|)$ by a process of partition refinement, as proved by [13] for undirected graphs, and detailed as applied to relations [2]. Thus, in linear time, one can merge all sets of lines which are equal. Note that after this process, the domination on attributes (*resp.* objects) will be a strict order.

The second tool we use extensively enables us to decide which lines (rows or columns) are properly included in another, or in other words determines a domination order. This can be done using the tripartite directed graph introduced by Bordat [5]. Computing the transitive edges of this graph will result into the domination order on objects or attributes, depending on how the graph is initially defined [2]. Computing the transitive closure of a graph can be performed in the same time as Matrix Multiplication, with a time complexity of $O(n^{\alpha})$, where α is currently 2.376 [6]. However, the $O(n^{2.376})$ algorithm [6] for Matrix Multiplication is not often used, as it is difficult to implement. It is easy to compute the domination order in $O(nm)$ time, as each line can be compared to all the other lines in linear time.

The last step of our algorithm requires a transitive reduction which consists in removing all the transitivity edges of a partial order. This problem has the same time complexity as the equivalent problem of transitivity closure and can also be performed in the same time as matrix multiplication.

5 Algorithmic process

Our algorithm works in five simple steps:

1. Clarify the input relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$ into a relation $\mathcal{R}_c$ where no two lines (rows or columns) are identical in order to avoid redundancy.
2. Compute the domination relation $Dom_{\mathcal{A}}$ between attributes (*i.e.* decide which columns of $\mathcal{R}_c$ are included into which other columns).
3. Compute a new relation $\mathcal{R}_{ce}$, obtained by appending $Dom_{\mathcal{A}}$ to $\mathcal{R}_c$, and simplify $\mathcal{R}_{ce}$ into $\mathcal{R}_{ces}$ where no two rows are identical. (This simplification merges an attribute and an object whenever they are introduced by the same concept.)
4. Extract from $\mathcal{R}_{ces}$ the elements of $\mathcal{H}(\mathcal{R})$, whose intents are in fact the rows of $\mathcal{R}_{ces}$ and whose simplified labels are the labels of these rows in $\mathcal{R}_{ces}$.
5. Construct the Hasse diagram of $\mathcal{H}(\mathcal{R})$ from these intents.

Note that as objects and attributes play symmetric roles, the algorithm can dually use domination on objects instead of attributes. The choice may result from an unbalanced number of objects with respect to the number of attributes.

5.1 Clarifying $\mathcal{R}$ into $\mathcal{R}_c$

Some objects (resp. attributes) may have the same image set (resp. antecedent set) and will then appear in the same concepts and share the same introducer. To simplify this redundancy, we will then merge identical lines of $\mathcal{R}$ to obtain clarified relation $\mathcal{R}_c$. This can be done in linear $O(|\mathcal{R}|)$ time, as discussed in Subsection 4.2.

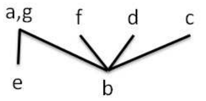
Example. In relation $\mathcal{R}$ of our running example, attributes a and g have the same antecedent set $\{1, 2, 3, 7\}$, objects 2 and 8 have the same image set $\{a, e, f\}$. The corresponding clarified relation $\mathcal{R}_c$ is presented. $\mathcal{R}_c$ and $\mathcal{R}$ have the same lattice and the same Galois sub-hierarchy.

$\mathcal{R}_c$	a,g	b	c	d	e	f
1	×		×	×	×	
2,8	×				×	×
3	×	×	×	×		×
4			×			×
5				×		
6			×	×		
7	×				×	

5.2 Computing $Dom_{\mathcal{A}}$ from $\mathcal{R}_c$

The domination relation on attributes $Dom_{\mathcal{A}}$ is computed using clarified relation $\mathcal{R}_c$ as input. $Dom_{\mathcal{A}}$ has been proven to be a sub-order of the Galois sub-hierarchy where only the elements having an attribute in their simplified label have been preserved [4]. As discussed in Subsection 4.2, this can be done in $O(|Attr|^{\alpha})$ or in $O(|\mathcal{A}|.|\mathcal{R}_c|)$ time.

Example. The domination order $Dom_{\mathcal{A}}$ of $\mathcal{R}_c$ is represented here as a sub-order of $\mathcal{H}(\mathcal{R})$. a and g have been grouped by the clarification process. Then b strictly dominates ag, f, d, and c: $Dom_{\mathcal{A}}(b) = \{ag, f, d, c, b\}$, and e strictly dominates ag: $Dom_{\mathcal{A}}(e) = \{ag, e\}$.



5.3 Constructing relation $\mathcal{R}_{ce}$ and its simplification $\mathcal{R}_{ces}$

We now compute relation $\mathcal{R}_{ce}$, which is the juxtaposition of $\mathcal{R}_c$ with $Dom_{\mathcal{A}}$. The formal definition of $\mathcal{R}_{ce} \subseteq (\mathcal{O} \cup \mathcal{A}) \times \mathcal{A}$ is as follows: $\forall x \in \mathcal{O}$, $\forall y \in \mathcal{A}$, $(x, y) \in \mathcal{R}_{ce}$ iff $(x, y) \in \mathcal{R}_c$, and $\forall y, z \in \mathcal{A}$, $(y, z) \in \mathcal{R}_{ce}$ iff $(y, z) \in Dom_{\mathcal{A}}$.

Now relation $\mathcal{R}_{ce}$ may have identical rows. As the input relation has already been clarified, this can only occur when an object has the same image set (in $\mathcal{R}_c$) as an attribute (in $Dom_{\mathcal{A}}$). We will merge these lines of $\mathcal{R}_{ce}$ to obtain a new relation $\mathcal{R}_{ces}$. We will show in the next section that this last process will associate the rows of $\mathcal{R}_{ces}$ with the elements of $\mathcal{H}(\mathcal{R})$.

This simplification, as the clarification of Step 1, can be obtained in linear time; however, the process now only compares objects with attributes. Note that the initial clarification into $\mathcal{R}_c$ could have been delayed and integrated into this step, but the more redundancies the initial relation contains, the more time the computation of $Dom_{\mathcal{A}}$ will require; a better average time complexity is thus obtained by separating these steps.

Example. $\mathcal{R}_c(3) = Dom_{\mathcal{A}}(b)$, so 3 and b are merged in $\mathcal{R}_{ces}$, as 5 with d, and 7 with e.

$\mathcal{R}_c$	a,g	b	c	d	e	f
1	×		×	×	×	
2,8	×				×	×
3	×	×	×	×		×
4			×			×
5				×		
6			×	×		
7	×				×	

$+$

$Dom_{\mathcal{A}}$	a,g	b	c	d	e	f
a,g	×					
b	×	×	×	×		×
c			×			
d				×		
e	×				×	
f						×

$=$

$\mathcal{R}_c + Dom_{\mathcal{A}} = \mathcal{R}_{ce}$

$\mathcal{R}_{ce}$	a,g	b	c	d	e	f
1	×		×	×	×	
2,8	×				×	×
3	×	×	×	×		×
4			×			×
5				×		
6			×	×		
7	×				×	
a,g	×					
b	×	×	×	×		×
c			×			
d				×		
e	×				×	
f						×

$\to$

$\mathcal{R}_{ce} \to \mathcal{R}_{ces}$

$\mathcal{R}_{ces}$	a,g	b	c	d	e	f
1	×		×	×	×	
2,8	×				×	×
4			×			×
6			×	×		
a,g	×					
3,b	×	×	×	×		×
c			×			
5,d				×		
7,e	×				×	
f						×

5.4 Extracting the elements of $\mathcal{H}(\mathcal{R})$ from $\mathcal{R}_{ces}$

We will now prove that the starting set of $\mathcal{R}_{ces}$ yields exactly the elements of $\mathcal{H}(\mathcal{R})$, because of our two-step merging process. Step 1 grouped together separately equivalent objects or equivalent attributes which trivially correspond to objects or attributes having the same introducer. Step 3 grouped together an object and an attribute whenever they have the same introducer, as proved in Theorem 3. Thus the labels of the rows of $\mathcal{R}_{ces}$ are the simplified labels of $\mathcal{H}(\mathcal{R})$, and for each row, its elements yield the intent of the corresponding concept, as proved in Theorem 4. No extra computation is thus needed for this step.

Example. The starting set of $\mathcal{R}_{ces}$ is: { {1}, {2,8}, {4}, {6}, {a,g}, {3,b}, {c}, {5,d}, {7,e}, {f} }. Its elements correspond exactly to the simplified label of the elements of $\mathcal{H}(\mathcal{R})$ presented in Figure 1. The rows represent the intents of these elements: for example, the complete labeling of the introducer of 2 would be ({2,8},{a,g,e,f}).

Theorem 3. *Given a relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$, the introducer of $x \in \mathcal{O}$ and the introducer of $y \in \mathcal{A}$ are the same if and only if $\mathcal{R}_{ce}(x) = \mathcal{R}_{ce}(y)$.*

Proof.
– Suppose concept C_{xy} is the introducer of both $x \in \mathcal{O}$ and $y \in \mathcal{A}$. The intent of C_{xy} is by definition $\mathcal{R}(x)$, which includes y. Let z be an attribute of $\mathcal{R}(x) = \mathcal{R}_{ce}(x)$ and C_z its introducer; as $(x, z) \in \mathcal{R}$, $C_{xy} \leq C_z$ and then Extent(C_{xy})⊆Extent(C_z) (Theorem 1); therefore, $(y, z) \in Dom_{\mathcal{A}}$ and so $z \in \mathcal{R}_{ce}(y)$; thus $\mathcal{R}_{ce}(x) \subseteq \mathcal{R}_{ce}(y)$. Let t be an attribute in $\mathcal{R}_{ce}(y)$ and C_t its introducer; $(y, t) \in Dom_{\mathcal{A}}$, which implies $C_t \geq C_{xy}$, and so the extent of C_t contains x, which implies $(x, t) \in \mathcal{R}$, *i.e.* $t \in \mathcal{R}_{ce}(x)$; thus $\mathcal{R}_{ce}(y) \subseteq \mathcal{R}_{ce}(x)$.
– Suppose $\mathcal{R}_{ce}(x) = \mathcal{R}_{ce}(y)$, *i.e.* $\mathcal{R}(x) = Dom_{\mathcal{A}}(y)$. For $z \in Dom_{\mathcal{A}}(y)$, $(x, z) \in$

$\mathcal{R}$, so the introducers of z and x are comparable: $C_z \geq C_x$ (Theorem 1); in particular, $C_y \geq C_x$. By definition of the domination relation, $\mathcal{R}^{-1}(y) \subseteq \mathcal{R}^{-1}(z)$ for all $z \in Dom_{\mathcal{A}}(y) = \mathcal{R}(x)$ and then, by definition of the ordering of $\mathcal{L}(\mathcal{R})$, $C_z \geq C_y$; the extent of C_y is included in the intersection of the extents of these C_z. As we are dealing with elements of a lattice, the extent of C_x is the intersection of the extents of all C_z, $z \in \mathcal{R}(x)$, which means C_x is the infimum (greatest lower bound) of these C_z. Finally, C_y is the minimum concept of a set of concepts of which C_x is the infimum, clearly $C_x = C_y$.
◇

Consequently, the final relation $\mathcal{R}_{ces}$ yields the elements of $\mathcal{H}(\mathcal{R})$:

Theorem 4. *The rows of $\mathcal{R}_{ces}$ are in a one-to-one correspondence with the elements of $\mathcal{H}(\mathcal{R})$. More precisely, each element of the starting set is the simplified label of the corresponding element of $\mathcal{H}(\mathcal{R})$ and its image set is the intent of this concept.*

Proof.
Each object or attribute has an associated introducer. Two objects (resp. attributes) have the same introducer if and only if their image sets (resp. antecedent sets) are equal; the corresponding lines of $\mathcal{R}$ have been merged in $\mathcal{R}_c$. In the other hand, by Theorem 3, an object and an attribute have the same introducer if and only if $\mathcal{R}_{ce}(x) = \mathcal{R}_{ce}(y)$; the corresponding lines of $\mathcal{R}_{ce}$ have been merged in $\mathcal{R}_{ces}$. As a consequence, the starting sets of $\mathcal{R}_{ces}$ are the simplified labels of the elements of $\mathcal{H}(\mathcal{R})$. Their image sets are the corresponding intents: for $x \in \mathcal{O}$, $\mathcal{R}_{ces}(x) = \mathcal{R}(x)$ is the intent of the introducer of x; for $y \in \mathcal{A}$, $\mathcal{R}_{ces}(y) = \{z \in \mathcal{A} \,|\, (y, z) \in Dom_{\mathcal{A}}\}$ which corresponds to the intent of the introducer of y.
◇

Note that the use of $Dom_{\mathcal{A}}$ gives the intent sets of the elements of $\mathcal{H}(\mathcal{R})$. The use of $Dom_{\mathcal{O}}$ instead would have given the extent sets. The use of both $Dom_{\mathcal{A}}$ and $Dom_{\mathcal{O}}$, as proposed in [4], is less efficient for computing the elements of $\mathcal{H}(\mathcal{R})$.

5.5 Constructing the Hasse diagram of $\mathcal{H}(\mathcal{R})$

Now all we have left to do is construct the Hasse diagram of $\mathcal{H}(\mathcal{R})$ by constructing the ordering by inclusion on the intents. This can be done in $O((|\mathcal{O}|+|\mathcal{A}|)^{\alpha})$ time by removing all transitivity edges, as discussed in Subsection 4.2.

6 The algorithm

```
Algorithm HERMES
Input: binary relation $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$.
Output: $\mathcal{H}(\mathcal{R})$
    Compute clarified relation $\mathcal{R}_c$;    //merge all identical lines
    Compute relation $Dom_{\mathcal{A}}$;          //determine column inclusions in $\mathcal{R}_c$
    $\mathcal{R}_{ce}$=$\mathcal{R}_c$+$Dom_{\mathcal{A}}$;                  //simple juxtaposition
    Simplify $\mathcal{R}_{ce}$ into $\mathcal{R}_{ces}$;         //merge all identical rows
    Extract the elements of $\mathcal{H}(\mathcal{R})$;  //select the starting set of $\mathcal{R}_{ces}$
    Order the elements of $\mathcal{H}(\mathcal{R})$;    //by inclusion on their image sets.
```

The complexity of the algorithm is bounded by Steps 2 and 5 with a time in $O((|\mathcal{O}|+|\mathcal{A}|).|\mathcal{R}|)$ or $O((|\mathcal{O}|+|\mathcal{A}|)^{\alpha})$, depending on the chosen implementation.

7 Specialized input: chordal-bipartite relations

A special class of relations should be mentioned in this context: relations which correspond to 'chordal-bipartite graphs', which are bipartite graphs containing no chordless cycle of length six or more. This is a superclass of the relations which have a planar lattice, but the lattice of chordal-bipartite relations remains of polynomial size [8].

Relations whose corresponding bipartite graph is chordal-bipartite can be re-ordered so that their matrix becomes 'Γ-free'. A Γ in a matrix is a sub-matrix on 4 elements, with a unique zero in the right-hand lower corner (*i.e.* in matrix M, there is a pair h,i of rows, $h < i$, and a pair j,k of columns, $j < k$, such that M(h,j)=M(h,k)=M(i,j)=1 and M(i,k)=0).

This Γ-free form is obtained by computing a 'Double Lexical Ordering' (DLO) [17]. A DLO is an ordering of the matrix such that the binary 'words' read from bottom to top for columns are in increasing lexical order, and likewise for rows, the binary words read from right to left are in increasing order from bottom to top. In the example below, column a has word 1001**0** which is smaller than the word of b, which is 0000**1**, and likewise the word of object 2, 00**0**_01_ is smaller than the word of object 3, 00**1**_01_.

Any matrix can be re-ordered to be DLO, and this is re-ordering can be done in time $O(min\{m\,logn,\,n^2\})$ [19, 23]. The DLO matrix is Γ-free if and only if the relation is chordal bipartite [17].

When a relation is in such a DLO and Γ-free form, it is easy to compute $Dom_{\mathcal{A}}$: take each attribute from left to right; for each attribute y, let x be the first object (from top to bottom) in the column of y (*i.e.* the first x such that $(x, y) \in \mathcal{R}$); then y dominates exactly the attributes z which are to its right and that are on row x (*i.e.* $(x, z) \in \mathcal{R}$).

This a consequence of the *DLO* and Γ-free form: in a DLO matrix, a given column can not be included in any column to its left; and in a Γ-free matrix, if w is the first row with a one in column y, for any column z at the right of y

which has a one in the row of w, if column y is not included in column z, as the rows of y above w all have zeros, this might only be because of a row x after w with a one in column y and a zero in column z, *i.e.* because of a Γ in the matrix formed by rows w and x, and columns y and z.

Example. The following matrix is ordered in a double lexical fashion and is Γ-free. Attribute a is processed first; its first one is on row 1, so a dominates all the attributes to its right which has a one on row 1: a dominates d. Attribute b is processed next; its first one is on row 5, which has ones at the right of b for c, d and e, b dominates c, d and e. Attribute c: highest one in row 3, c dominates e. Attribute d: highest one in row 1, no one at the right, no domination. Attribute e is last and therefore can dominate no other attribute.

$\mathcal{R}$	a	b	c	d	e
1	1			1	
2					1
3			1		1
4	1			1	1
5		1	1	1	1

When relation $\mathcal{R}$ is chordal-bipartite, $\mathcal{R}_{ces}$ can then be constructed in $O(n^2)$. We conjecture that the Hasse diagram can be extracted at no extra cost.

8 Conclusion

We have presented a new, simple, and more efficient algorithm, Hermes, for building the Galois sub-hierarchy of a relation. It would be interesting to compare its running time in practice to that of the other known algorithms; we conjecture that Hermes will run faster in most cases.

Algorithm Hermes could be remodeled into an incremental algorithm, which may prove interesting for on-line applications such as updating hierarchies in object-oriented languages.

Acknowledgement

The authors thank Lhouari Nourine for fruitful discussions on the construction of Galois sub-hierarchies.

References

1. Arévalo G., Berry A., Huchard M., Perrot G., and Sigayret A.: *Comparison of performances of Galois sub-hierarchy-building algorithms.* Proc. of ICFCA'07, LNCS 4390, p. 166-180. 2007.
2. Berry A., Bordat J.-P., and Sigayret A.: *A local approach to concept generation.* Ann. Math. Artificial Intelligence 49, p.117-136. 2007.
3. Berry A., Huchard M., McConnell R.M., Spinrad J.P.: *Efficiently Computing a Linear Extension of the Sub-hierarchy of a Concept Lattice.* Proc. of ICFCA'05. 2005

4. Berry A., and Sigayret A.: *Maintaining Class Membership Information.* Workshop Maspeghi, proc. OOIS'02 (Conference on Object-Oriented Information Systems). 2002.
5. Bordat J-P.: *Calcul pratique du treillis de Galois d'une correspondance.* Mathématiques, Informatique et Sciences Humaines, 96, p. 31-47. 1986.
6. Coppersmith D., and Winograd S.: *Matrix multiplication via arithmetic progressions.* Proc. 9th Annual ACM Symposium on Theory of Computing, p.1-6. 1987.
7. Dicky H., Dony C., Huchard M., and Libourel T.: *Ares, adding a class and restructuring inheritance hierarchies.* Proc. BDA'95, p. 25-42. 1995.
8. Eschen E., Pinet N., Sigayret A.: *Consecutive-ones: handling lattice planarity efficiently.* Proc. CLA'07. 2007.
9. Ganter B., Wille R.: *Formal Concept Analysis: Mathematical Foundations.* Springer. 1999.
10. Godin R., Mili H.: *Building and Maintaining Analysis-Level Class Hierarchies Using Galois Lattices.* Proc. OOPSLA'93, p.394-410. 1993.
11. Godin R., Chau T-T.: *Comparaison d'algorithmes de construction de hiérarchies de classes*, Journal L'OBJET, 5:3-4. 1999.
12. Hitzer P.: *Default reasoning over Domains and Concepts Hierarchies.* Proc. KI'04, LNCS 3238, p.351-365. 2004.
13. Hsu W.-L., and Ma T.-H.: *Substitution decomposition on chordal graphs and its applications.* SIAM Journal on Computing, 28, p.1004-1020. 1999.
14. Huchard M., Dicky H., and Leblanc H.: *Galois Lattice as a Framework to specify Algorithms Building Class Hierarchies.* Theoretical Informatics and Applications, 34, p. 521-548. 2000.
15. Huchard M., Leblanc H.: *Computing Interfaces in Java.* ASE, p.317-320. 2000.
16. Leblanc H. *Sous-hiérarchies de Galois: un modèle pour la construction et l'évolution des hiérarchies d'objets* (in French). PHD thesis, Université Montpellier 2. 2000.
17. Lubiw A.: *Doubly lexical orderings of matrices.* SIAM Journal on Computing, 16(5), p.854-879. 1987.
18. Osswald R., and Petersen W.: *Introduction of Classification from Linguistic Data.* Proc. ECAI'02, Workshop FCAKDD, p.75-84. 2002.
19. Paige R., Tarjan R.E.: *Three partition refinement algorithms.* SIAM J. Comput. 16(6), p.973-989. 1987.
20. Petersen W.: *A Set-Theoretical Approach for the Induction of Inheritance Hierarchies.* Electronic Notes in Theoretical in Computer Science, 51. 2001.
21. Ryssel U., Ploennigs J., Kabitzsch K.: *Extraction of feature models from formal contexts.* SPLC Workshops (15th Int. Software Product Line Conference). 2011.
22. Sigayret A.: *Data mining: une approche par les graphes* (in French). PHD thesis, Université Blaise Pascal (Clermont-Ferrand, France). 2002.
23. Spinrad J.-P.: *Doubly lexical ordering of dense 0-1 matrices.* Information Processing Letters, 45(5), p.229–235. 1993.
24. http://conexp.sourceforge.net/download.html, release 1.3. © S.A. Yevtushenko & al. 2000-2006.