CLA 2007
Fifth International Conference on Concept Lattices and Their Applications

Montpellier, October 24-26, 2007

Web site : http ://www.lirmm.fr/cla07

1 Preface

The Concept lattice and their Applications conference series began as a response to the increasing use of concept lattices in practical applications. These proceedings reflect the diversity of interest in concept lattices by the scientific community with papers ranging from the analysis of algorithms in data mining to teaching mathematics.

CLA is structured as a triple peer-review process : conference paper, presentation and extended journal paper. 48 papers were submitted to CLA2007 and 24 papers were selected for presentation. All papers were peer reviewed by at least 3 members of the program committee with the final decision on acceptance determined from the reviews by the program chairs. Based on the presentation of the final papers, outstanding contributions from this volume will be invited to submit an extended version of their conference paper for a special volume on concept lattice applications for the International Journal of General Systems. These papers will again be peer-reviewed.

We would particularly like to thank the program committee members and reviewers who gave of their time and we extend our thanks to the authors who contributed to this volume.
Finally, we would also like to thank our sponsors, Lirmm, CNRS, Montpellier II University, and Languedoc Roussillon Region.

Jean Diatta, Peter Eklund and Michel Liquiere

2 Table of Contents

3 Organization

Steering Committee	
Radim Belohlavek	U.S.A
Sadok Ben Yahia	Tunisia
Engelbert Mephu Nguifo	France
Vaclav Snasel	Czech Republic

Program co-chairs	
Jean Diatta	France
Peter Eklund	Australia

Program committee	
Radim Belohlavek	U.S.A
Claudio Carpineto	Italy
Samir Elloumi	Tunisia
Mohamed Gammoudi	Tunisia
Robert Godin	Canada
Marianne Huchard	France
Ali Jaoua	Qatar
Stanislav Krajci	Slovakia
Sergei Kuznetsov	Russia
Bruno Leclerc	France
Michel Liquiere	France
Mondher Maddouri	Tunisia
Rokia Missaoui	Canada
Amedeo Napoli	France
Sergei Obiedkov	Russia
Uta Priss	Scotland
Henri Ralambondrainy	France
Yahya Slimani	Tunisia
Gerd Stumme	Germany
Petko Valtchev	Canada
Vilem Vychodil	Czech Republic
Mohammed Zaki	U.S.A

External reviewers	
Benjarath Phoophakdee	U.S.A
Mohamed Quafafou	France

Organization Chair	
Michel Liquière	Lirmm, France

Organization Committee	
Celine Berger	Lirmm, France
Jean-Remy Falleri	Lirmm, France
Marianne Huchard	Lirmm, France
Chedy Raissi	Lirmm, France

Approximate clusters and biclusters

Boris Mirkin

Birkbeck College, University of London, UK,
mirkin@dcs.bbk.ac.uk

Abstract

In the talk, I review my results and related work on approximation of (weighted) graphs and digraphs - that is, rectangular or square similarity or flow matrices - by single clusters and biclusters.These are akin to maximum density subgraphs and spectral clusters and, as well, to some popular heuristic clustering algorithms. Theoretical and computational results will be presented. Approximate biclusters may be of interest as an extension of the formal concepts.I am going to show how approximate clusters can be used for further aggregating "clusters" of highly overlapping formal concepts to drastically reduce the numbers of "relaxed" formal concepts to study. Another development of interest is further mapping of the clusters to a taxonomic or phylogenetic hierarchy, that serves as an independent tool for verification of the clusters, with application to profiling and evolution analysis.

Closure operators and choice operators : a survey

Bernard Monjardet

CES, Université Paris 1,
Bernard.Monjardet@univ-paris1.fr

Abstract

In this talk I will give a overview on the connections between closure operators and choice operators and on related results. An operator on a finite set S is a map defined on the set P(S) of all the subsets of S. A closure operator is an extensive, isotone and idempotent operator. A choice operator c is a contracting operator ($c(A) \subseteq A$, for every $A \subseteq S$). Choice operators and their lattices have been very studied in the framework of the theory of the revealed preference in economics. A significant connection between closure operators and choice operators is the duality between anti-exchange operators (corresponding to convex geometries) and path-independent choice operators. More generally, there is a one-to-one correspondence between closure operators and choice operators.

Some references

Aleskerov F., Bouyssou D., Monjardet B., (2007.) Utility maximisation, choice and preference, (Studies in Economic Theory 16), Springer-Verlag.
Caspard N., Monjardet B., (2004). Some lattices of closure systems Discrete Mathematics and Theoretical Computer Science, 6, 163-190. Danilov V., Koshevoy G. A., (2006). Choice functions and extending operators, preprint.
J. Demetrovics, G. Hencsey, L. Libkin, I.B. Muchnik, (1992) On the interaction between closure operations and choice functions with applications to relational databases, Acta Cybernetica 10 (3), 129 – 139.
Echenique (2007), Counting combinatorial choice rules, Games and Economic Behavior 58 (2007), 231-245.
Johnson, M. R., Dean, R.A (1996): "An Algebraic Characterization of Path Independent Choice Functions," Third International Meeting of the Society for Social Choice and Welfare, Maastricht, TheNetherlands.
Koshevoy G. A., (1999) Choice functions and abstract convex geometries. Mathematical Social Sciences, 38(1), 35-44.
Monjardet B., Raderinirina V. (2001) The duality between the anti-exchange closure operators and the path independent choice operators on a finite set. Mathematical Social Sciences, 41(2), 131-150.

Concept Graphs as Semantic Structures for Contextual Judgment Logic

Rudolf Wille

Technische Universität Darmstadt, Fachbereich Mathematik,
Schloßgartenstr. 7, D–64289 Darmstadt; wille@mathematik.tu-darmstadt.de

Abstract. This paper presents a mathematization of the *philosophical doctrine of judgments* as an extension of the mathematization of the philosophical doctrine of concepts developed in Formal Concept Analysis. The chosen approach was strongly stimulated by J. F. Sowa's theory of *conceptual graphs.* The mathematized conceptual graphs, called *concept graphs*, are mathematical semantic structures based on formal contexts and their formal concepts; those semantic structures are viewed as formal judgments in the underlying *Contextual Judgment Logic.* In this paper concept graphs are systematically built up starting with *simple concept graphs* in section 2 and continuing with *existential concept graphs* in section 3, with *implicational and clausal concept graphs* in section 4, and finally with generalizations of concept graphs in section 5. Examples are illustrating the different types of concept graphs.

Contents

1 Semantic Structures for Contextual Judgment Logic

"Contextual Logic" has grown out of attemps to "restructure" lattice theory and mathematical logic (see [Wi82], [Wi96], [Wi97], [Pr98]). These attempts were stimulated by the german scholar Hartmut von Hentig with his charge to *restructure scientific disciplines* which he explains as follows:

> "The restructuring of scientific disciplines within themselves become more and more necessary to make them better learnable, mutually available, and criticizable in more general surroundings, also beyond disciplinary competence. This restructuring may and must be performed by general patterns of perceptions, thought, and action of our civilization." ([He74], p.33f.)
> For this: "Sciences have to examine their disciplinarity, and this means: to uncover their unconscious purposes, to declare their conscious purposes, to select and to adjust their means according to those purposes, to explain possible consequences comprehensibly and publicly, and to make accessible their ways of scientific finding and their results by the every-day language." ([He74], p.136f.)

Restructuring lattice theory has been started in 1979 by mathematizing concepts and concept hierarchies which led to the notions of "formal context" and "concept lattice" (see [Wi82]). A *formal context* was defined as a triple (G, M, I) where G is a set, the elements of which are called "objects", M is a set, the elements of which are called "attributes", and $I \subseteq G \times M$ is a binary relation for which $(g, m) \in I$ (also written: gIm) is read: "the object g has the attribute m". A *formal concept* of (G, M, I) was then defined as a pair (A, B) with $A \subseteq G$ and $B \subseteq M$ satisfying:

$$A = \{g \in G \mid \forall m \in B : gIm\}(=: B') \text{ and } B = \{m \in M \mid \forall g \in A : gIm\}(=: A').$$

A and B are called the *extent* and the *intent* of the formal concept (A, B), respectively. The set $\mathfrak{B}(G, M, I)$ of all formal concepts of a given formal context (G, M, I) carries an order relation $\leq$ defined by $(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2$ $(\Leftrightarrow B_1 \supseteq B_2)$ so that $(\mathfrak{B}(G, M, I), \leq)$ becomes a complete lattice, the so-called *concept lattice* of (G, M, I) which is structured by the following $\bigwedge$−operation and $\bigvee$−operation:

$$\bigwedge_{t\in T}(A_t, B_t) := (\bigcap_{t\in T} A_t, (\bigcup_{t\in T} B_t)''), \qquad \bigvee_{t\in T}(A_t, B_t) := ((\bigcup_{t\in T} A_t)'', \bigcap_{t\in T} B_t).$$

A comprehensive introduction to the rich theory of concept lattices is presented in the monograph *"Formal Concept Analysis: Mathematical Foundation"* [GW99a].

Restructuring mathematical logic has been started in the early 1990s and first presented at the conference on *"logic and algebra"* held in Pontignano (Siena) in April 1994 (see [Wi96]). The restructuring approach was based on the traditional philosophical logic which is founded on "the three essential main functions of thinking - *concepts*, *judgments*, and *conclusions*" ([Ka88], p.6) and therefore, on the elementary level, presented in three parts: the doctrine of concepts, the doctrine of judgments, and the doctrine of conclusions. These doctrines are essential, since human thinking is based on concepts as basic units of thought, on judgments as assertional combinations of concepts, and on conclusions as entailments between judgments. For mathematizing concepts, judgments, and conclusions, they shall be primarily understood as *semantic structures* which are basic for logical thinking.

Mathematizing the doctrine of concepts, using a contextual approach, has been already performed to a great extent in developing *Formal Concept Analysis* (cf. [GW99a], in particular: [GW99b],[Wi00]),[Ga05]). Therefore, this paper continues to present the *mathematization of the doctrine of judgments* which builds up a *Contextual Judgment Logic* based on developments in Formal Concept Analysis (cf. [Wi01],[Wi03]). The chosen approach was strongly stimulated by J. F. Sowa's theory of conceptual graphs [So84] since those graphs can be understood as semantic structures which represent logical judgments. The mathematized conceptual graphs, called *concept graphs*, are mathematical semantic structures based on formal contexts and their formal concepts (cf. [Wi97],[Wi02]); those semantic structures are considered as formal judgments in the underlying Contextual Judgment Logic. How concept graphs can be systematically introduced and analysed is described in the next sections: *simple concept graphs* in section 2, *existential concept graphs* in section 3, *implicational and clausal concept graphs* in section 4, and generalizations of concept graphs, in particular *concept graphs with local negations* in section 5.

2 Simple Concept Graphs and Their Conceptual Contents

Each step of the presented development of concept graphs shall start with an example of a judgment represented graphically by a conceptual graph as standardized by John Sowa (cf. [So92]). Those judgments are deduced from the following statement written by Charles S. Peirce ([Pe92], p.114):

"Mathematics ... is the only one of the sciences which does not concern itself to inquire what the actual facts are, but studies hypotheses exclusively."

To obtain an example of a *simple conceptual graph*, we consider the judgment: "The science mathematics studies the hypothesis $2^{\aleph_0} = \aleph_1$" (called "continuum hypothesis"). This judgment may be represented by the simple conceptual graph shown in Fig. 1. In that graph, "science" and "hypothesis" name concepts, while "mathematics" and "$2^{\aleph_0} = \aleph_1$" name objects which fall under the concepts "science" and "hypothesis", respectively; furthermore, the relational concept "study" links the science "mathematics" with the hypothesis "$2^{\aleph_0} = \aleph_1$".

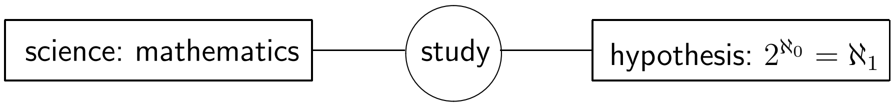


Fig. 1. Example of a simple conceptual graph

The example shows that judgments may join plain concepts with relational concepts so that a mathematization of judgments has to offer besides formal concepts also "relation concepts". How this has been performed and further developed shall be explained in the rest of this section (cf. [Wi04], pp. 53 – 55).

A *power context family* is a sequence $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ of formal contexts $\mathbb{K}_k := (G_k, M_k, I_k)$ with $G_k \subseteq (G_0)^k$ for $k = 1, 2, \ldots$. The formal concepts of $\mathbb{K}_k$ with $k = 1, 2, \ldots$ are called *relation concepts*, because they represent k-ary relations on the object set G_0 by their extents.

A *relational graph* is a structure (V, E, ν) consisting of two disjoint sets V and E together with a map $\nu : E \to \bigcup_{k=1,2,\ldots} V^k$; the elements of V and E are called *vertices* and *edges*, respectively, and $\nu(e) = (v_1, \ldots, v_k)$ is read: $v_1, \ldots, v_k$ are the *adjacent vertices* of the *k-ary edge* e ($|e| := k$ is the *arity* of e; the arity of a vertex is defined to be 0). Let $E^{(k)}$ be the set of all elements of $V \cup E$ of arity k ($k = 0, 1, 2, \ldots$).

A *simple concept graph* of a power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ with $\mathbb{K}_k := (G_k, M_k, I_k)$ for $k = 0, 1, 2, \ldots$ is a structure $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ for which

- (V, E, ν) is a relational graph,
- $\kappa: V \cup E \to \bigcup_{k=0,1,2,\ldots} \underline{\mathfrak{B}}(\mathbb{K}_k)$ is a mapping such that $\kappa(u) \in \underline{\mathfrak{B}}(\mathbb{K}_k)$ for all $u \in E^{(k)}$,
- $\rho: V \to \mathfrak{P}(G_0) \backslash \{\emptyset\}$ is a mapping such that $\rho(v) \subseteq Ext(\kappa(v))$ for all $v \in V$ and, furthermore, $\rho(v_1) \times \cdots \times \rho(v_k) \subseteq Ext(\kappa(e))$ for all $e \in E$ with $\nu(e) = (v_1, \ldots, v_k)$;
- in general, $Ext(\mathfrak{c})$ denotes the extent of the formal concept $\mathfrak{c}$.

It is convenient to consider the mapping ρ not only on vertices but also on edges: for all $e \in E$ with $\nu(e) = (v_1, \ldots, v_k)$, let $\rho(e) := \rho(v_1) \times \cdots \times \rho(v_k)$.

A *subgraph* of a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ is a concept graph $\mathfrak{G}_s := (V_s, E_s, \nu_s, \kappa_s, \rho_s)$ for which $V_s \subseteq V$, $E_s \subseteq E$, $\nu_s = \nu|_{E_s}$, $\kappa_s = \kappa|_{V_s \cup E_s}$, and $\rho_s = \rho|_{V_s}$. The *union* and *intersection* of subgraphs $\mathfrak{G}_t := (V_t, E_t, \nu_t, \kappa_t, \rho_t)$ $(t \in T)$ of a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ are defined by

$$\bigcup_{t\in T} \mathfrak{G}_t := (\bigcup_{t\in T} V_t, \bigcup_{t\in T} E_t, \bigcup_{t\in T} \nu_t, \bigcup_{t\in T} \kappa_t, \bigcup_{t\in T} \rho_t),$$

$$\bigcap_{t\in T} \mathfrak{G}_t := (\bigcap_{t\in T} V_t, \bigcap_{t\in T} E_t, \bigcap_{t\in T} \nu_t, \bigcap_{t\in T} \kappa_t, \bigcap_{t\in T} \rho_t).$$

Lemma 1 *The union and intersection of subgraphs of a concept graph $\mathfrak{G}$ is always a subgraph of $\mathfrak{G}$ again.*

From the background knowledge coded in a power context family $\vec{\mathbb{K}}$, two types of material inferences shall be made formally explicit: Let $k = 0, 1, 2, \ldots$;

1. *object implications*: for $A, C \subseteq G_k$, $\mathbb{K}_k$ satisfies $A \to C$ if $A^{I_k} \subseteq C^{I_k}$ and,
2. *concept implications*: for $\mathfrak{B}, \mathfrak{D} \subseteq \mathfrak{B}(\mathbb{K}_k)$, $\mathbb{K}_k$ satisfies $\mathfrak{B} \to \mathfrak{D}$ if $\bigwedge \mathfrak{B} \leq \bigwedge \mathfrak{D}$.

The *formal implications* $A \to C$ and $\mathfrak{B} \to \mathfrak{D}$ give rise to a closure system $\mathcal{C}(\mathbb{K}_k)$ on $\mathbb{S}^{imp}(\mathbb{K}_k) := \{(g, \mathfrak{b}) \in G_k \times \mathfrak{B}(\mathbb{K}_k) \mid g \in Ext(\mathfrak{b})\}$ consisting of all subsets Y of $\mathbb{S}^{imp}(\mathbb{K}_k)$ which have the following property:

(P_k) If $A \times \mathfrak{B} \subseteq Y$ and if $\mathbb{K}_k$ satifies $A \to C$ and $\mathfrak{B} \to \mathfrak{D}$ then $C \times \mathfrak{D} \subseteq Y$.

For $k = 1, 2, \ldots$, the $\mathbb{K}_k$-*conceptual content* $C_k(\mathfrak{G})$ of a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ of a power context family $\vec{\mathbb{K}}$ is defined as the closure of

$$\{(\vec{g}, \kappa(e)) \mid e \in E^{(k)} \text{ and } \vec{g} \in \rho(e)\}$$

with respect to the closure system $\mathcal{C}(\mathbb{K}_k)$;
the $\mathbb{K}_0$-*conceptual content* $C_0(\mathfrak{G})$ of $\mathfrak{G}$ is defined as the closure of

$$\{(g, \kappa(v)) \mid v \in V \text{ and } g \in \rho(v)\} \cup$$
$$\{(g_i, (G_0, G_0^{I_0})) \mid \exists((g_1, \ldots, g_k), \mathfrak{c}) \in C_k(\mathfrak{G}) \text{ with } g_i \in \{g_1, \ldots, g_k\}\}$$

with respect to the closure system $\mathcal{C}(\mathbb{K}_0)$. Then

$$C(\mathfrak{G}) := C_0(\mathfrak{G}) \,\dot\cup\, C_1(\mathfrak{G}) \,\dot\cup\, C_2(\mathfrak{G}) \,\dot\cup\, \ldots$$

is called the ($\vec{\mathbb{K}}$-)*conceptual content* of the concept graph $\mathfrak{G}$.

The defined conceptual contents give rise to an *information* (*quasi-*) *order* $\lesssim$ on the set of all concept graphs of a power context family: A concept graph $\mathfrak{G}_1 := (V_1, E_1, \nu_1, \kappa_1, \rho_1)$ is said to be *less informative* (*more general*) than a concept graph $\mathfrak{G}_2 := (V_2, E_2, \nu_2, \kappa_2, \rho_2)$ (in symbols: $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$) if

$$C_k(\mathfrak{G}_1) \subseteq C_k(\mathfrak{G}_2) \text{ for } k = 0, 1, 2, \ldots;$$

$\mathfrak{G}_1$ and $\mathfrak{G}_2$ are called *equivalent* (in symbols: $\mathfrak{G}_1 \sim \mathfrak{G}_2$) if $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$ and $\mathfrak{G}_2 \lesssim \mathfrak{G}_1$ (i.e., $C_k(\mathfrak{G}_1) = C_k(\mathfrak{G}_2)$ for $k = 0, 1, 2, \ldots$). The set of all equivalence classes of concept graphs of a power context family $\vec{\mathbb{K}}$ together with the order induced by the quasi-order $\lesssim$ is a *complete lattice* denoted by $\widetilde{\Gamma}(\vec{\mathbb{K}})$.

3 Existential Concept Graphs and Their Conceptual Contents

To obtain an example of an *existential conceptual graph*, we modify the judgment of section 2 as follows: "The science mathematics studies hypotheses". Logically equivalent is the judgment: "There exists at least one hypothesis studied by the science of mathematics". This judgment may be represented by the existential conceptual graph shown in Fig. 2.

The example shows that judgments may embody existentially quantified variables which are usually indicated by letters like x, y, z (sometimes they are replaced by a so-called "coreference link"). The mathematization of existential conceptual graphs whose variables are from a variable set X can be based on "free X-extensions" of a power context family. Such mathematization generalizes the approach of section 2 so that it becomes a wider range of applications (cf. [Wi04], pp. 55 – 57).

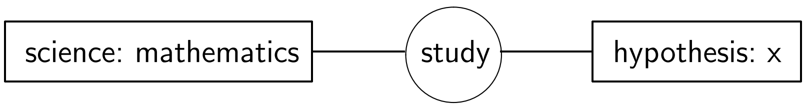


Fig. 2. Example of an existential conceptual graph

For a set X of variables, an X-*interpretation* into a set G_0 with $G_0 \cap X = \emptyset$ is defined as a mapping $\chi : G_0 \cup X \to G_0$ with $\chi(g) = g$ for all $g \in G_0$; the set of all X-interpretations into G_0 is denoted by $B(X, G_0)$. The *free* X*-extension* of the power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ with $\mathbb{K}_k := (G_k, M_k, I_k)$ for $k = 0, 1, 2, \ldots$ and $G_0 \cap X = \emptyset$ is defined as a power context family $\vec{\mathbb{K}}[X] := (\mathbb{K}_0[X], \mathbb{K}_1[X], \mathbb{K}_2[X], \ldots)$ for which

– $\mathbb{K}_0[X] := (G_0[X], M_0[X], I_0[X])$ with $G_0[X] := G_0 \cup X$, $M_0[X] := M_0$, $I_0[X] := I_0 \cup (X \times \{m \in M_0 \mid \{m\}^{I_0} \neq \emptyset\})$,
– $\mathbb{K}_k[X] := (G_k[X], M_k[X], I_k[X])$ $(k = 1, 2, \ldots)$ with $G_k[X] := \{(u_1, \ldots, u_k) \in G_0[X]^k | \exists \chi \in B(X, G_0) : (\chi(u_1), \ldots, \chi(u_k)) \in G_k\}$, $M_k[X] := M_k$, and $(u_1, \ldots, u_k) I_k[X] m :\iff \exists \chi \in B(X, G_0) : (\chi(u_1), \ldots, \chi(u_k)) I_k m$.

$\vec{\mathbb{K}}[X]$ is called an *existential power context family.*

For defining existential concept graphs, the surjective $\bigwedge$-homomorphisms $\pi_k^X : \underline{\mathfrak{B}}(\mathbb{K}_k[X]) \to \underline{\mathfrak{B}}(\mathbb{K}_k)$ $(k = 0, 1, 2, \ldots)$ are needed which are determined by

$$\pi_k^X(A, B) := (A \cap G_k, (A \cap G_k)^{I_k}) \text{ for } (A, B) \in \underline{\mathfrak{B}}(\mathbb{K}_k[X]).$$

An *existential concept graph* of a power context family $\vec{\mathbb{K}}$ is defined as a concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ of a free X-extension $\vec{\mathbb{K}}[X]$ for which an X-interpretation χ into G_0 exists such that $\mathfrak{G}^\chi := (V, E, \nu, \kappa^\chi, \rho^\chi)$ with $\kappa^\chi(u) := \pi_k^X(\kappa(u))$ and $\rho^\chi(v) := \chi(\rho(v))$ is a concept graph of $\vec{\mathbb{K}}$; χ is then called an X-interpretation *admissible on* $\mathfrak{G}$. For a fixed variable set X, $\mathfrak{G}$ is more precisely named an existential concept graph of $\vec{\mathbb{K}}$ *over* X.

Lemma 2 *The subgraphs of an existential concept graph over* X *are existential concept graphs over* X*, too.*

The *conceptual content* of an existential concept graph $\mathfrak{G}_X$ of a power context family $\vec{\mathbb{K}}$ is defined as the conceptual content of $\mathfrak{G}_X$ understood as a concept graph of the free X-extension $\vec{\mathbb{K}}[X]$. To clarify this, it is helpful to show how variables give rise to object implications of the relational contexts $\mathbb{K}_k[X]$ as indicated in the following lemma:

Lemma 3 *Let* $\mathbb{K}_k[X] := (G_k[X], M_k[X], I_k[X])$ *with* $k \in \{1, 2, \ldots\}$ *be a relational context of an existential power context family* $\vec{\mathbb{K}}[X]$*; furthermore, let* α *be a map of* $G_0 \cup X$ *into itself satisfying* $\alpha(g) = g$ *for all* $g \in G_0$*. Then* $\mathbb{K}_k[X]$ *has the object implications* $\{(\alpha(u_1), \ldots, \alpha(u_k))\} \longrightarrow \{(u_1, \ldots, u_k)\}$ *with* $u_1, \ldots, u_k \in G_0 \cup X$.

For a permutation π of the variable set X, let α_π be the map of $G_0 \cup X$ into itself with $\alpha_\pi(g) = g$ for all $g \in G_0$ and $\alpha_\pi(x) = \pi(x)$ for all $x \in X$. Then we obtain the object implication $\{(\alpha_\pi(u_1), \ldots, \alpha_\pi(u_k))\} \longrightarrow \{(u_1, \ldots, u_k)\}$ with $u_1, \ldots, u_k \in G_0 \cup X$. Together with the corresponding object implication for π^{-1}, this yields that changing variables according to a permutation of X in a (k-ary) object of $\mathbb{K}_k[X]$ does not change the intension of that object.

An existential concept graph $\mathfrak{G}_1 := (V_1, E_1, \nu_1, \kappa_1, \rho_1)$ is said to be *less informative* (*more general*) than $\mathfrak{G}_2 := (V_2, E_2, \nu_2, \kappa_2, \rho_2)$ (in symbols: $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$) if

$C_k(\mathfrak{G}_1) \subseteq C_k(\mathfrak{G}_2)$ for $k = 0, 1, 2, \ldots$; $\mathfrak{G}_1$ and $\mathfrak{G}_2$ are called *equivalent* (in symbols: $\mathfrak{G}_1 \sim \mathfrak{G}_2$) if $\mathfrak{G}_1 \lesssim \mathfrak{G}_2$ and $\mathfrak{G}_2 \lesssim \mathfrak{G}_1$ (i.e., $C_k(\mathfrak{G}_1) = C_k(\mathfrak{G}_2)$ for $k = 0, 1, 2, \ldots$). The set of all equivalence classes of existential concept graphs of a power context family $\vec{\mathbb{K}}$ over a fixed set X of variables together with the order induced by the quasi-order $\lesssim$ is an *ordered set* denoted by $\widetilde{\Gamma}(\vec{\mathbb{K}}; X)$.

4 Implicational and Causal Concept Graphs

For representing exactly Peirce's judgment "mathematics studies hypotheses exclusively", we have to generalize existential conceptual graphs further to *implicational conceptual graphs*. This becomes clear when we consider an equivalent formulation of Peirce's judgment, namely: "If mathematics studies a proposition then mathematics studies a hypothesis". A representation of this judgment by an implicational conceptual graph is pictured in Fig. 3.

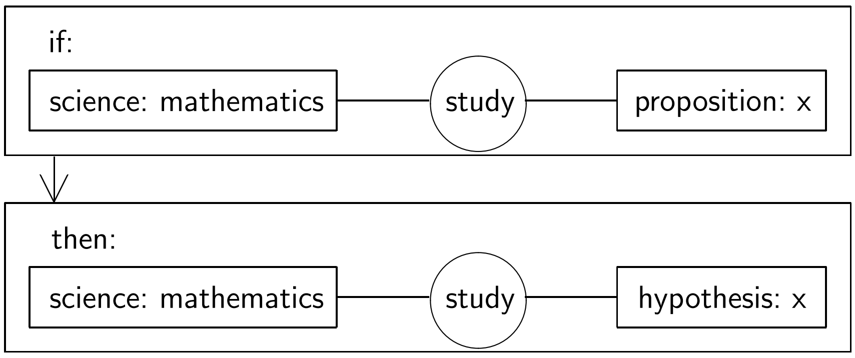


Fig. 3. Example of an implicational conceptual graph

The example shows an implicational judgment in which the premise and the conclusion contain the same variable x; this indicates that the proposition x is, more precisely, a hypothesis. The mathematization of implicational conceptual graphs who are composed by two subgraphs representing a premise and a corresponding conclusion, respectively, can be viewed as a generalization of existential concept graphs (cf. [Wi04], pp. 57 – 59).

An *implicational concept graph* of a power context family $\vec{\mathbb{K}}$ is defined as an existential concept graph $\mathfrak{G} := (V, E, \nu, \kappa, \rho)$ of $\vec{\mathbb{K}}$ over a variable set X with a designated pair $(p\mathfrak{G}, c\mathfrak{G})$ of subgraphs such that

1. $\mathfrak{G}$ is the union of $p\mathfrak{G}$ and $c\mathfrak{G}$, and
2. each X-interpretation admissible on $p\mathfrak{G}$ is also admissible on $c\mathfrak{G}$ (and hence on $\mathfrak{G}$ too).

$p\mathfrak{G} \to c\mathfrak{G}$ may be written instead of $\mathfrak{G}$; the subgraphs $p\mathfrak{G}$ and $c\mathfrak{G}$ are called the *premise* and the *conclusion*, resp.

For an existential concept graph $\overline{\mathfrak{G}}$ of a power context family $\vec{\mathbb{K}}$ over a variable set X, the formal context $\mathbb{K}(X; \overline{\mathfrak{G}}) := (B(X, G_0), Sub(\overline{\mathfrak{G}}), \rhd)$ is defined where

– the object set $B(X, G_0)$ consists of all X-interpretations into the object set G_0 of the formal context $\mathbb{K}_0$ in $\vec{\mathbb{K}}$,
– the attribute set $Sub(\overline{\mathfrak{G}})$ is the set of all subgraphs of $\overline{\mathfrak{G}}$,
– $\chi \rhd \mathfrak{G}$ means that the X-interpretation χ is admissible on the subgraph $\mathfrak{G}$ of $\overline{\mathfrak{G}}$.

Proposition 1 *$\{\mathfrak{G}_s \mid s \in S\} \to \{\mathfrak{G}_t \mid t \in T\}$ is an attribute implication of $\mathbb{K}(X;\overline{\mathfrak{G}})$ if and only if $\bigcup_{s\in S}\mathfrak{G}_s \to \bigcup_{t\in T}\mathfrak{G}_t$ is an implicational concept graph of $\vec{\mathbb{K}}$ over X.*

Proposition 2 *$\mathbb{K}(X;\overline{\mathfrak{G}}) := (B(X,G_0), Sub(\overline{\mathfrak{G}}), \rhd)$ is always a formal context of which all extents are non-empty attribute extents. Conversely, let $\mathbb{K} := (G,M,I)$ be a clarified formal context of which all extents are non-empty attribute extents; then $\mathbb{K}$ is isomorphic to the clarified context of the formal context $\mathbb{K}(\{x\};\overline{\mathfrak{G}}) := (B(\{x\},G), Sub(\overline{\mathfrak{G}}), \rhd)$ where $\overline{\mathfrak{G}} := (V,E,\nu,\kappa,\rho)$ is the existential concept graph of the power context family $\vec{\mathbb{K}} := (\mathbb{K})$ over $\{x\}$ with $V := M$, $E := \emptyset$, $\nu := \emptyset$, $\kappa(m) := \mu m$, and $\rho(m) := \{x\}$.*

Corollary 1 *The concept lattices $\underline{\mathfrak{B}}(\mathbb{K}(X;\overline{\mathfrak{G}}))$ are up to isomorphism the concept lattices of formal contexts.*

Implicational conceptual graphs can even be generalized to clausal conceptual graphs in which the conclusion consists of a disjunction of propositions (cf. [Wi04], pp. 59 – 60). An example of a clausal conceptual graph is shown in Fig. 4.

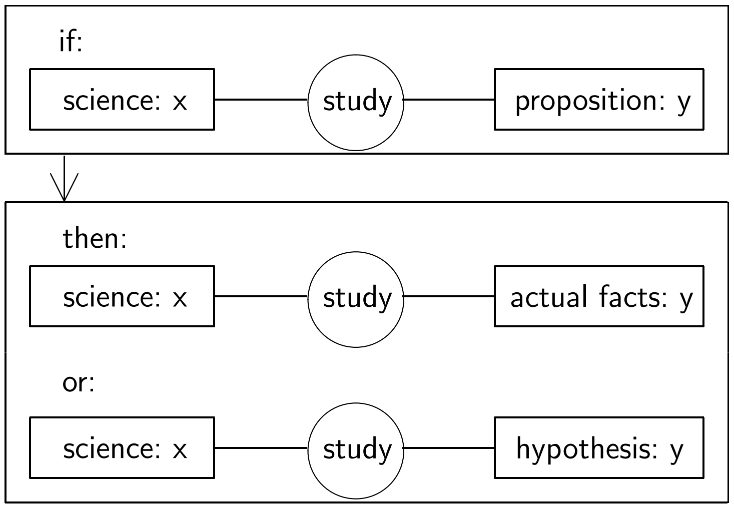


Fig. 4. Example of a clausal conceptual graph

A *clausal concept graph* of a power context family $\vec{\mathbb{K}}$ is defined as an existential concept graph $\mathfrak{G} := (V,E,\nu,\kappa,\rho)$ of $\vec{\mathbb{K}}$ over a variable set X with a designated pair $(p\mathfrak{G}, \{c_t\mathfrak{G} \mid t \in T\})$ consisting of a subgraph $p\mathfrak{G}$ of $\mathfrak{G}$ and a set $\{c_t\mathfrak{G} \mid t \in T\}$ of subgraphs of $\mathfrak{G}$ such that

1. $\mathfrak{G}$ is the union of $p\mathfrak{G}$ and all the $c_t\mathfrak{G}$ with $t \in T$, and
2. each X-interpretation admissible on $p\mathfrak{G}$ is also admissible on at least one $c_t\mathfrak{G}$ with $t \in T$.

$p\mathfrak{G} \to \bigvee_{t\in T} c_t\mathfrak{G}$ may be written instead of $\mathfrak{G}$; the subgraphs $p\mathfrak{G}$ and $c_t\mathfrak{G}$ $(t \in T)$ are called the *premise* and the *disjunctive conclusions*, resp. For subsets A and B of the attribute set M, $\bigwedge A \to \bigvee B$ is an *attribute clause* of $\mathbb{K}$ if $g \in A^I$ always implies gIm for at least one $m \in B$.

Proposition 3 *Let $\overline{\mathfrak{G}}$ be an existential concept graph of a power context family $\vec{\mathbb{K}}$ over a variable set X and let $\mathfrak{G}_s$ $(s \in S)$ and $\mathfrak{G}_t$ $(t \in T)$ be subgraphs of $\overline{\mathfrak{G}}$. Then $\bigwedge\{\mathfrak{G}_s \mid s \in S\} \to \bigvee\{\mathfrak{G}_t \mid t \in T\}$ is an attribute clause of the formal context $\mathbb{K}(X;\overline{\mathfrak{G}})$ if and only if $\bigcup_{s\in S}\mathfrak{G}_s \to \bigvee_{t\in T}\mathfrak{G}_t$ is a clausal concept graph of $\vec{\mathbb{K}}$ over X.*

Proposition 2 and 3 show that the theory of clausal concept graphs is essentially equivalent to the theory of attribute clauses of formal contexts. The advantage of this equivalence is that many results about attribute clauses can be transferred to clausal concept graphs which substantially enriches the research on Contextual Judgment Logic.

Corollary 2 $\mathfrak{G}_\emptyset \to \bigvee_{t\in T} \mathfrak{G}_t$ *is a clausal concept graph of* $\vec{\mathbb{K}}$ *over* X *if and only if for all* X*-interpretations* χ *into* G_0 *there exists a* $t_\chi \in T$ *such that* χ *is admissible on* $\mathfrak{G}_{t_\chi}$.

5 Generalizations of concept graphs

Concepts and concept graphs form a comprehensive core for a *semantics of Contextual Logic*. Although such a semantics offers a great variety of support for logical thinking, there is still the desire to use further conceptual structures. Here only two types of such structures shall be discussed briefly so that readers get at least an idea about the richness which still has to be explored.

As a first type of generalized graphs we want to consider *conceptual graphs with local negation*. For this, we start again with an example deduced from Peirce's statement cited at the beginning of section 2; the example is shown in Fig. 5 which presents an implicational conceptual graph with local negation. The diagram can

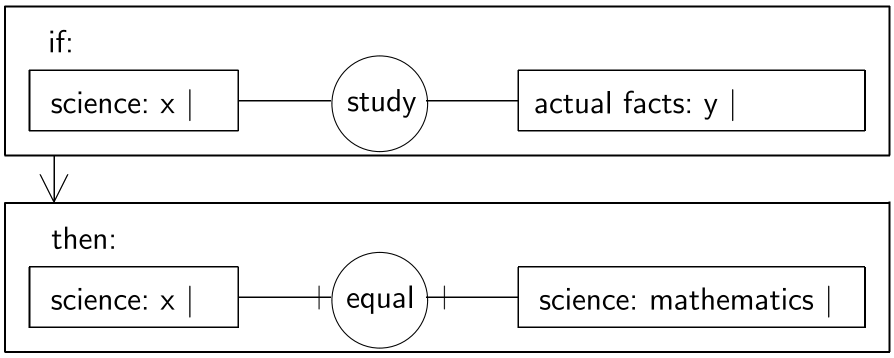


Fig. 5. Example of an implicational conceptual graph with local negation

be read: "If a science studies actual facts then this science is not mathematics" or "mathematics is a science which does not study actual facts". The additional vertical strokes which divide the spaces after the colon in the rectangular boxes allow to represent the negation of the object-concept-relation (see [Wi02]). The vertical strokes cutting the horizontal lines joining the rectangular boxes with the equal-circle indicate the negation of the relation "equal"; such localization means that "x is not mathematics".

Graphs with local negation have been introduced as "*protoconcept graphs*" in [Wi02]. An extensive elaboration of the logic system of those protoconcept graphs with their syntax and semantics can be found in [Kl05]. The logic system of *concept graphs with negation* (and its relationship to predicate logic) which can be understood as a mathematization of a large fragment of Sowa's theory of conceptual graphs, has been impressively worked out and published in the Springer Lecture Notes in Artificial Intelligence [Da03].

As a second type of generalized graphs we want to mention *conceptual graphs with a modal component*. Nested conceptual graphs may be understood to have a modal component. This becomes clear through a mathematical representation of nested conceptual graphs by *triadic concept graphs* which have been invented in [Wi98]. The discussion in that paper clarifies that one should consider not only nestings, but also subdivisions with overlappings. As an example for this, a triadic concept graph is shown which represents a diatonic modulation from C-major to A-major with its constitutive chord overlappings. Triadic concept graphs are based on a triadic power context family $\vec{\mathbb{K}} := (\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2, \ldots)$ with $\mathbb{K}_k := (G_k, M_k, B, I_k)$ and $G_k \subseteq G_0^k$ $(k = 0, 1, 2, ...)$ where B is always a set of *modalities*. It can be shown that the triadic concept graphs of a triadic power context family always form a complete lattice with respect to the generalization order. It turns out that the generalization order may be differently defined, depending on the assumed background knowledge, respectively (cf. [GW00]).

A *concept graph with subdivision* is a mathematical structure derived from a triadic power context family. The aim of introducing concept graphs with subdivision is to represent modal information mathematically. This has been demonstrated in [SW03] by an example, namely by a comparison of the two famous paintings: the "Darmstädter Madonna" and the "Dresdner Madonna". Based on the notion of a *conceptual content*, the concept graphs with subdivision of a triadic power context family has been proved to form a complete lattice with respect to the *information order* (cf. [SW03]).

Finally, it shall be pointed out that a *conceptual theory and methodology of semantic structures*, named *"semantology"*, are under development from which the theory and practice of concept graphs and their generalizations, and therefore of Contextual Judgment Logic could benifit. The initial paper [GW06] discusses, from the view of Peirce's classification of sciences, a *three-fold semantics* of conceptual knowledge: the mathematical, the philosophical and an application-oriented semantics. Examples from *Formal Concept Analysis* are considered. The second paper [EW07] extends the discussion of the three-fold semantics to *Conceptual Knowledge Processing* by using the extensive analysis of methods in Formal Concept Analysis and Contextual Logic presented in [Wi06]. A special case-study about applications of semantology in music is offered in [WW07]. For understanding how mathematical methods can be applied in the real world, the relationship between mathematics and concept analysis is analysed in [Wi07], in particular by the three-fold semantics of concept analysis in Conceptual Knowledge Representation.

References

[Da03] F. Dau: The logic system of concept graphs with negation. Dissertation, TU Darmstadt 2003. Shaker Verlag, Aachen 2003.

[EW07] P. Eklund, R. Wille: Semantology as basis for Conceptual Knowledge Processing. In: S. O. Kuznetsov, St. Schmidt (eds.): *Formal Concept Analysis. ICFCA 2007.* LNAI **4390**. Springer, Heidelberg 2007, 18–38.

[Ga05] B. Ganter: Contextual attribute logic of many-valued contexts. In: B. Ganter, R. Wille, K. E. Wolff (eds.): *Formal Concept Analysis. State-of-the-Art Survey.* LNAI **3626**. Springer, Heidelberg 2005, 101-113.

[GW99a] B. Ganter, R. Wille: *Formal Concept Analysis: mathematical foundations.* Springer, Heidelberg 1999; German version: Springer, Heidelberg 1996.

[GW99b] B. Ganter, R. Wille: Contextual attribute logic. In: W. Tepfenhart, W. Cyre (eds.): *Conceptual structures: standards and practices.* LNAI **1640**. Springer, Heidelberg 1999, 377–388.

[GW06] P. Gehring, R. Wille: Semantology: basic methods for knowledge representations. In: H. Schärfe, Pascal Hitzler, Peter Øhrstrøm (eds.): *Conceptual structures: inspiration and application.* LNAI **4068**. Springer, Heidelberg 2006, 215–228.

[GW00] B. Groh, R. Wille: Lattices of triadic concept graphs. In: B. Ganter, G. Mineau (eds.): *Conceptual structures: logical, linguistic, and computational issues.* LNAI **1867**. Springer, Heidelberg 2000, 332-341.

[He74] H. von Hentig: Magier oder Magister? Über die Einheit der Wissenschaften im Verständigungsprozeß. Suhrkamp Taschenbuch **207**. Suhrkamp, Frankfurt 1974.

[Ka88] I. Kant: Logic. Dover, Mineola N.Y. 1988.

[Kl05] J. Klinger: The logic system of protoconcept graphs. Dissertation, TU Darmstadt 2005. Shaker Verlag, Aachen 2005.

[Pe92] Ch. S. Peirce: Reasoning and the logic of things. Edited by K. L. Ketner; with an introduction by K. L. Ketner and H. Putnam. Havard University Press, Cambridge 1992.

[Pr98] S. Prediger: Kontextuelle Urteilslogik mit Begriffsgraphen. Ein Beitrag zur Restrukturierung der mathematischen Logik. Dissertation. TU Darmstadt 1998. Shaker, Aachen 1998.

[SW03] L. Schoolmann, R. Wille: Concept graphs with subdivision: a semantic approach. In: A. de Moor, W. Lex, B. Ganter (eds.): *Conceptual structures for knowledge creation and communication.* LNAI **2746**. Springer, Heidelberg 2003, 271–281.

[So84] J. F. Sowa: Conceptual structures: Information processing in mind and machine. Adison-Wesley, Reading 1984.

[So92] J. F. Sowa: Conceptual graphs summary. In: T. E. Nagle, J. A. Nagle, L. L. Gerholz, P. W. Eklund (eds.): *Conceptual structures: current research and practice.* Ellis Horwood, 1992, 3–51.

[Wi82] R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets.* Reidel, Dordrecht-Boston 1982, 445–470.

[Wi96] R. Wille: Restructuring mathematical logic: an approach based on Peirce's pragmatism. In: A. Ursini, P. Agliano (eds.): *Logic and algebra.* Marcel Dekker, New York 1996, 267-281.

[Wi97] R. Wille: Conceptual Graphs and Formal Concept Analysis. In: D. Lukose, H. Delugach, M. Keeler, L. Searle, J. F. Sowa (eds.): *Conceptual Structures: Fulfilling Peirce's Dream.* LNAI **1257**. Springer, Heidelberg 1997, 290–303.

[Wi98] R. Wille: Triadic concept graphs. In: M. L. Mugnier, M. Chein (eds.): *Conceptual structures: theory, tools and applications.* LNAI **1453**. Springer, Heidelberg 1998, 194-208.

[Wi00] R. Wille: Boolean Concept Logic. In: B. Ganter, G. Mineau (eds.): *Conceptual structures: logical, linguistic, and computational issues.* LNAI **1867**. Springer, Heidelberg 2000, 317-331.

[Wi01] R. Wille: Boolean Judgment Logic. In: H. S. Delugach, G. Stumme (eds.): *Conceptual structures: broadening the base.* LNAI **2120**. Springer, Heidelberg 2001, 115-128.

[Wi02] R. Wille: Existential concept graphs of power context families. In: U. Priss, D. Corbett, G. Angelova (eds.): *Conceptual structures: integration and interfaces.* LNAI **2393**. Springer, Heidelberg 2002, 382–395.

[Wi03] R. Wille: Conceptual content as information - basics for Conceptual Judgment Logic. In: A. de Moor, W. Lex, B. Ganter (eds.): *Conceptual structures for knowledge creation and communication.* LNAI **2746**. Springer, Heidelberg 2003, 1–15.

[Wi04] R. Wille: Implicational concept graphs. In: K. E. Wolff, H. Pfeiffer, H. Delugach (eds.): *Conceptual structures at work.* LNAI **3127**. Springer, Heidelberg 2004, 52–61.

[Wi06] R. Wille: Methods of Conceptual Knowledge Processing. In: R. Missaoui, J. Schmid (eds.): *Formal Concept Analysis. ICFCA 2006.* LNAI **3874**. Springer, Heidelberg 2006, 1–29.

[Wi07] R. Wille: Formal Concept Analysis as applied lattice theory. In: R. Bělohlvek, S. Ben Yahia, E. Mephu Nguifo (eds.): *Concept lattices and their applications. CLA 2006.* LNAI . Springer, Heidelberg 2007, .

[WW07] R. Wille, R. Wille-Henning: Towards a semantology of music. In: U. Priss, S. Polovina (eds.): *Conceptual structures: knowledge architectures for smart applications.* LNAI . Springer, Heidelberg 2007, .

Some variations on Alan Day's algorithm for calculating canonical basis of implications

Vincent Duquenne

CNRS-ECP6, Université Pierre et Marie Curie,
175 rue du Chevaleret, 75013 Paris, France
duquenne@math.jussieu.fr

Abstract. Two variations of an algorithm by Alan Day for reducing a list of implications regarding redundancy are given, with a new simple justification. All three algorithms have the property that the list can be reduced *in place* -at no extra memory cost- that will be useful for large applications and databases.

Keywords: basis of implications, closure operator, reduction in place, redundancy.

Introduction

Many years after having "introduced implication basis into FCA" (as noted in [GW99 p.94]) which dates back to the fall of 1983 (see [GD84-86], [G84-87], [D84-87]), it is surprising if not hard to acknowledge that we didn't learn a lot more on "their intimacy" in the meantime, despite many interesting papers using or revisiting them. In that respect, a special attention should be paid to one of the last papers by Alan Day ([Day92]), who spent several months of his spare time to clarifying the interest of *Lattice Theory* for putting *databases* into *canonical forms*, by decomposing them through *functional dependencies* and *relation schemes*. This was linked with other works (among which [W95]) that made precise the connections with functional dependencies ([MA83]) -classical in databases and AI-, to the FCA community. More recently we had ([D&A01], [O&D03-07], [V&D03-07]) to device some variations around Alan's algorithm and some pain to explain how they work which we now do.

Let A be a (finite) set of *attributes*, $L:=\{X_i \rightarrow Y_i \mid i\in I, X_i, Y_i \subseteq A\}$ be a *list of attribute implications*, and let consider the two closure operators:

(1) *L-closure*, $X \mapsto L(X)$ for all $X\subseteq A$, that is defined by reiteration of $X^L:=X\cup\bigcup\{Y_i \mid X_i\rightarrow Y_i\in L, X_i\subseteq X\}$ up to reaching a fixpoint (hence finiteness…), namely $L(X):=X^{L\ldots L}=(X^{L\ldots L})^L$ which is the *consequence* of X, and similarly,

(2) *L-saturation* $X \mapsto L°(X)$ for all $X\subseteq A$, $L°(X):=X^{L°\ldots L°}=(X^{L°\ldots L°})^{L°}$ defined by reiteration of $X^{L°}:=X\cup\bigcup\{L(X_i) \mid X_i\rightarrow Y_i\in L, X_i\subseteq X, L(X_i)\neq L(X)\}$.

The L-saturation L°(X) is a *restricted consequence* of X for "what is already known for smaller premises than X out of X's L-closed class". Notice how it is a bit harder to handle since dependent of the L-closure due to the condition $L(X_i)\neq L(X)$ that requires to have L(X) at hand when calculating L°(X) and will thus slow down algorithms.

An implication is *full* whenever its conclusion is L-closed, L itself is said to be *full* when $L:=\{X_i \rightarrow L(X_i) \mid i \in I, X_i \subseteq A\}$, which could be taken as a strong hypothesis, but is very natural when the input is a context. To avoid repeating $X_i \rightarrow Y_i \cup X_i$ all the time in the sequel, we always suppose that $L:=\{X_i \rightarrow Y_i \mid i \in I, X_i, Y_i \subseteq A\}$ is such that $X_i \subset Y_i$ ($i \in I$) and sometimes that L is full, specifying wherever results and algorithms can be extended to families of non-full implications. We freely mix together our original terminology and denotations with more established ones in FCA [GW99].

Now, $X=L°(X)$ is called *L-saturated* for short. A saturated subset $L°(X)$ is called *L-quasi-closed* when not closed $L°(X) \neq L(X)$ –some authors did differently…-, in which case $L(X)$ is called *essential L-closed* (or *meet-essential element* of the $\cap$-semi-lattice of L-closed subset, see [D84-87, D91]). $L°(X)$ is called *L-pseudo-closed* when $\subseteq$-minimal L-quasi-closed in $\{Y \subseteq A \mid L(Y)=L(X)\}$. The set $B_L:=\{X \rightarrow L(X) \mid X$ L-pseudo-closed$\}$ is called the *canonical basis* of L (*saturated* moreover, if necessary, sometimes *Guigues-Duquenne basis* or – a new comer- *stem basis* in FCA's folklore).

The main result in [GD84-86] states that B_L is a minimal set of implications inferring L -for usual propositional calculus, or so-called *Armstrong rules*- and that moreover any such minimal family is in one-one correspondence with B_L through a natural construction, hence the name *canonical saturated basis* [D84-87 p.225] that was first chosen. Many authors revisited these notions introducing their own denotations. Here, we will not come back on that, but focus on some properties of *quasi / pseudo-closed* subsets that should be extended for bettering the algorithms.

It is now part of the folklore (see the above references) that:

Lemma 1. For a list L, a subset $H \subseteq L$ defines the same closure operator (has the same set of consequences) as L iff H has the same canonical basis as L iff for every L-pseudo-closed X there is at least one $(X_i \rightarrow Y_i) \in H$ for which $X_i \subseteq X \subset L(X)=H(X_i)$.

It provided the characterizations formulated in early drafts of [D84-87]:

Lemma 2. For a closure operator L() on A (or a list L of implications...):
1. $X=L°(X)$ iff $L(X_i) \subset X$ for all $X_i \subseteq A$ s. t. $X_i \subset X$ and $L(X_i) \subset L(X)$.
2. $X=L°(X)$ iff $L(X_i) \subset X$ for all L-pseudo-closed X_i s. t. $X_i \subset X$ and $L(X_i) \subset L(X)$.
3. X is L-pseudo-closed or L-closed iff $L(X_i) \subseteq X$ for all L-pseudo-closed $X_i \subset X$.

Remark. (2) reduces (1) to L-pseudo-closed. (3) gives a recursive definition of L-pseudo-closed subsets, that [G84-87], [GW99] took as a starting definition instead.

Now when starting from a list of implications, it will be enough to rewrite (1) as:

Lemma 3. For a list $L:=\{X_i \rightarrow Y_i \mid i \in I, X_i \subset Y_i \subseteq A\}$, $X=L°(X)$ iff $L(X_i) \subset X$ for all $(X_i \rightarrow Y_i) \in L$ such that $X_i \subset X$ and $L(X_i) \subset L(X)$.

Our aim is to complete these basic properties for deriving either the canonical or some arbitrary basis of a list L, by dropping out redundant implications one at a time.

Reducing lists of full implications

Lists of full implications will be somehow easier to deal with thanks to the simple:

Lemma 4. For a full list of implications $L:=\{X_i\to Y_i=L(X_i)\,|\,i\in I,\ X_i\subset Y_i\subseteq A\}$, $(X_i\to Y_i)\in L$ and $H:=L\backslash\{X_i\to Y_i\}$,
1. $H(X_i)\neq L(X_i)$ iff
2. $L°(X_i)$ is L-pseudo-closed and there is no other $(X_k\to Y_k)\in L$ with $L°(X_k)=L°(X_i)$.

Proof. Since $\{X_i\to Y_i\in L,\ X_i\subseteq X,\ L(X_i)\neq L(X)\}\subseteq H\subset L$, notice that

(*) for $X\subseteq A$ with $L(X)=L(X_i)$, $X\subseteq L°(X)\subseteq H(X)\subseteq L(X)$ holds.

Suppose (2) fails. Case 1: $L°(X_i)$ not-quasi-closed hence L-closed $L°(X_i)=L(X_i)$ implies $H(X_i)=L(X_i)$ by (*). Case 2: $L°(X_i)$ L-quasi-closed not pseudo-closed implies the existence of a L-pseudo-closed Z with $Z\subset L°(X_i)\subset L(Z)=L(X_i)$, and by Lemma 1 the existence of at least one $(X_k\to Y_k)\in L$ with $L°(X_k)=Z$ so that $H(X_i)\supseteq L(X_k)=L(X_i)$ hence $H(X_i)=L(X_i)$. Case 3: $L°(X_i)$ pseudo-closed and the existence of some $(X_k\to Y_k)\in L$ with $X_k\neq X_i$ and $L°(X_k)=L°(X_i)$ implies $H(X_i)=H(X_k)=L(X_i)$. Conversely, suppose that (2) holds: since $L°(X_i)$ is L-quasi-closed and minimal in its L-closed class for this property, for any $(X_k\to Y_k)\in L$ such that $X_k\subset L°(X_i)$, $L(X_k)\subset L°(X_i)$ must hold by Lemma 3, which implies $L°(X_i)=H(L°(X_i))$, so that by (*) it comes that $X_i\subseteq L°(X_i)=H(L°(X_i))\subseteq H(H(X_i))=H(X_i)$, hence $H(X_i)=L°(X_i)\neq L(X_i)$, by isotony of H() and $L°(X_i)$ is L-pseudo-closed.

Algorithm 1.

Input a full family $L:=\{X_i\to Y_i=L(X_i)\,|\,i\in I,\ X_i\subset Y_i\subseteq A\}$.
Output: canonical basis $B_L:=\{X_k\to Y_k\,|\,k\in K,\ X_k$ L-pseudo-closed$\}$ (or an arbitrary one).

```
1. For $i\in I$
2.    $L=L\backslash\{X_i\to Y_i\}$   /drop it out /
3.    $X=L(X_i)$          /see below Amendment 1/
4.    If $X\neq Y_i$ Then
5.      $L=L\cup\{X\to Y_i\}$  /restore it when $X_i$ was the (last)/
6.    Endif            /generator of a pseudo-closed/
7. Endfor
```

We have used this algorithm in GLAD [D83-96] for years, specially in preparing [D&Al01], [O&D03-07], [V&D03-07]. It has been independently conceived in [R07].

Remarks. The negation of Lemma 4 (1) provides a simple criteria for dropping out redundant implications $X_i \rightarrow Y_i$ in L: when $H(X_i)=L(X_i)$. Notice that otherwise, $H(X_i)=L°(X_i)$ is so to say automatically delivered L-pseudo-closed thanks to fullness. Moreover, to get a basis with smaller premises replace 5 by `(5' L=L∪{X`$_i$`→Y`$_i$`})`.

The morality of this procedure is that L-pseudo-closed are generated by either their *single* generator or *last examined* generator whenever several are existing in L, which is another explanation -out of their recursive nature- for their difficulty to be reached.

A main feature of this algorithm is that the reduction can be done in place. The price is to suppose the list L full, which takes … full benefit of transitivity and keeps tracks of implication consequences by isotony after they have been dropped out. The bonus are that the painful part (statement 3) reduces in time as implications are dropped out. There is no post-processing to get the pseudo-closed and this can be used to extract a basis made of the original implications, while preserving the redundant ones by permutation of L, separating L in two areas basis / redundant implications.

Amendment 1. Statements 3-4 in Algorithm 1 can be replaced by the following:

```
3'. X=L-conditional(X_i,Y_i,L,Restore)
4'. If (Restore=true) Then
```

where L-conditional is a function that reiteratively calculates $L(X_i)$ but cancels the calculus as soon as (if ever) the criteria $X=Y_i$ is reached within the iterative loop (returning Restore=false), and returns $L(X_i)$ otherwise (with Restore=true). As many implications are redundant in practice, this usually will save time.

Hence, in any circumstances where L is naturally full or can be made full at small price –which is quite often the case in FCA when the input is a context- this will avoid to calculate the L-saturation of the X_is by applying the definition and dealing continuously with both the painful restrictions $L(X_k) \neq L(X_i)$ and deadly reiteration. Fullness provides the clarity and efficiency of this simple algorithm.

Reducing non-full implications

Now, in the context of databases and AI, it may be the case that dependencies are expressed by non-full implications. For instance, see the new developments in [B06] that promote *canonical direct basis* for which the L-closure / saturation do not require reiteration, but are always reached with a single scan of the basis in construction.

For reducing such lists of implications some specific properties are required:

Lemma 5. Let $L:=\{X_i \rightarrow Y_i \mid i \in I,\ X_i \subset Y_i \subseteq A\}$ be a list, $(X_i \rightarrow Y_i) \in L$ and $H:=L \setminus \{X_i \rightarrow Y_i\}$, suppose moreover that $H(X) \subset L(X)$ for some $X \subseteq A$, then:

1. $X \subseteq X \cup X_i \subset X \cup Y_i \subseteq L(X)$ and $L(X) \supseteq L(X_i)$, and
2. $H(X) \supseteq X_i$, and
3. $H(X)$.not.$\supseteq Y_i$ holds.

Proof. (1): $H(X) \subset L(X)$ implies that $X_i \rightarrow Y_i$ must take part in the iterative calculus of $L(X)$, so that $X \subseteq X \cup X_i \subset X \cup Y_i \subseteq L(X)$. By isotony and idempotence of $L()$, this implies $L(X) \subseteq L(X \cup X_i) \subseteq L(L(X)) = L(X)$, hence $L(X) = L(X \cup X_i) \supseteq L(X_i)$ must hold. (2): By contraposition. Since $X \subseteq H(X) \subset L(X)$ and $L \backslash H = \{X_i \rightarrow Y_i\}$ hold, $H(X)$.not.$\supseteq X_i$ implies $H(X) = L(X)$, a contradiction. (3): Similarly, $H(X) \supseteq Y_i \supset X_i$ implies $H(X) \cup X_i = H(X) \cup Y_i$, so that by contraposition of (1), $H(H(X)) = L(H(X))$, but $H(H(X)) = H(X)$ and $L(H(X)) = L(X)$, hence $H(X) = L(X)$ should hold, a contradiction, so that $H(X)$.not.$\supseteq Y_i$ holds as asserted.

This should be made a little bit more precise and gives some indications where the premises $X \subset A$ of non-redundant implications for which $H(X) \subset L(X)$ are, a potentiality that will be made clearer by the following:

Lemma 6. For a list $L := \{X_i \rightarrow Y_i \mid i \in I, X_i \subset Y_i \subseteq A\}$, $(X_i \rightarrow Y_i) \in L$ and $H := L \backslash \{X_i \rightarrow Y_i\}$,
1. either $H(X_i) \supseteq Y_i$, in which case $H(X_i) = L(X_i)$, $H() = L()$ and $X_i \rightarrow Y_i$ is L-redundant,
2. or $H(X_i)$.not.$\supseteq Y_i$, in which case $H(X_i) \subset L(X_i)$ and $H(X_i) = L°(H(X_i))$ is L-quasi-closed and $X_i \subseteq L°(X_i) \subseteq H(X_i) = L°(H(X_i)) \subset L(X_i)$ holds, that moreover collapses $L°(X_i) = H(X_i)$ whenever $L°(X_i)$ is L-pseudo-closed and there is no other $(X_k \rightarrow Y_k) \in L$ with $L°(X_k) = L°(X_i)$ and Y_k.not.$\subseteq L°(X_i)$.

Proof. (1): $H(X_i) \supseteq Y_i$ implies $H(X_i) = L(X_i)$ by contraposition of Lemma 5.3. Suppose that there exists some $X \subset A$ for which $H(X) \subset L(X)$, this implies $H(X) \supseteq X_i$ by Lemma 5.2, hence $H(X) \supseteq H(X_i) = L(X_i) \supseteq Y_i$ holds by isotony of $H()$, hence $H(X) = L(X)$ by contraposition of Lemma 5.3, a contradiction, so that $H() = L()$, and $X_i \rightarrow Y_i$ is therefore redundant in L. (2): Let $H(X_i)$.not.$\supseteq Y_i$ holds. $X_i \subset Y_i \subseteq L(X_i)$ and $X_i \subseteq H(X_i) \subseteq L(X_i)$ imply that $H(X_i) = L(X_i)$ implies $H(X_i) \supseteq Y_i$, a contradiction, hence $H(X_i) \subset L(X_i)$ must hold. Suppose now that $H(X_i)$ is not L-quasi-closed. By Lemma 3 there must exist some $(X_k \rightarrow Y_k) \in L$ for which $X_k \subseteq H(X_i)$, $L(X_k)$.not.$\subseteq H(X_i)$ and such that $L(X_k) \subset L(H(X_i)) = L(X_i)$. $L(X_k) \subset L(X_i)$ implies $L(X_k)$.not.$\supseteq L(X_i)$, hence $H(X_k) = L(X_k)$, by contraposition of Lemma 5.1. By isotony and idempotence of $H()$, $X_k \subseteq H(X_i)$ implies $X_k \subseteq H(X_k) \subseteq H(H(X_i)) = H(X_i)$, so that $L(X_k) \subseteq H(X_i)$, a contradiction, $H(X_i)$ is thus L-quasi-closed. $X_i \subseteq L°(X_i) \subseteq H(X_i) \subset L(X_i)$ follows from (*) in Lemma 4's proof. Last, $L°(X_i)$ L-pseudo-closed and there is no other $(X_k \rightarrow Y_k) \in L$ with $L°(X_k) = L°(X_i)$ and Y_k.not.$\subseteq L°(X_i)$ implies $L°(X_i) = H(L°(X_i))$, the above inequalities imply that $X_i \subseteq L°(X_i) = H(L°(X_i)) \subseteq H(X_i) \subset L(X_i)$ hold, but $L°(X_i) \supseteq X_i$ implies $H(L°(X_i)) \supseteq H(X_i)$ by isotony, so that $H(X_i) = L°(X_i)$ must hold.

Remark. Lemma 6 (1) provides an efficient criteria for dropping out most redundant implications in L, and in particular all those $(X_i \rightarrow Y_i) \in L$ with $L(X_i)$ not-essential L-closed (of which the L-closed classes contain no L-quasi-closed subset). Lemma 6 (2) guaranties that replacing $X_i \rightarrow Y_i$ in L by $H(X_i) \rightarrow H(H(X_i) \cup Y_i) = L(X_i)$ gives an equivalent set of implications with a new premise $H(X_i)$ that is made automatically L-quasi-closed, without having to calculate any saturation actually.

Together with Lemma 1 and reiteration, this can be used to produce a superset of the canonical basis, and provides a new simple justification of the algorithm that is given at the end of [Day92 p. 426], of which the original setting –although focused on functional dependencies- is somehow complex and quite algebraic in nature, since it is expressed with sophisticated constructions on semi-lattice congruence relations:

```
**Algorithm 2.** [Alan Day 1992 p.426]
Input a family $L := \{X_i \rightarrow Y_i \mid i \in I,\ X_i \subset Y_i \subseteq A\}$.
Output: an equivalent set of implications
$L := \{X_k \rightarrow Y_k \mid k \in K,\ X_k$ L-quasi-closed$\}$.
1. For $i \in I$
2.    $L = L \backslash \{X_i \rightarrow Y_i\}$        /drop it out /
3.    $X_i = L(X_i)$                    /see above Amendment 1/
4.    If $Y_i$.not.$\subseteq X_i$ Then
5.      $Y_i = L(X_i \cup Y_i)$    /restore non-redundant/
6.      $L = L \cup \{X_i \rightarrow Y_i\}$   /remade full implications/
7.    Endif
8. Endfor
```

Remark. As observed by Alan Day, getting the canonical basis ("*critical*" basis in his own terms) out of this new amended list requires a post-processing to sort it and check premises' minimal property in their L-closed classes. This is due to the fact that a non-minimal quasi-closed can be obtained and kept in the list while a smaller pseudo-closed in the same L-closure class makes it becoming redundant afterwards. Hence, the reduction process is dependent of the order taken for scanning through L, and there is no way to insure that quasi-closed are tested in a $\subseteq$-compatible order.

However, a main advantage of this algorithm is to drop out a lot of redundant implications (statements 2-4), although it cannot detect the L-pseudo-closed sets on the fly, that could be a handicap for real scale applications and huge databases.

This can be avoided by using their recursive characterization in Lemma 2.3:

Algorithm 3.

```
Input a family L:={$X_i$→$Y_i$|i∈I, $X_i$⊂$Y_i$⊆A}.
Output: basis $B_L$:={$X_k$→$Y_k$|k∈K, $X_k$ L-pseudo-closed}.
1.  For i∈I
2.    L=L\{$X_i$→$Y_i$}    /drop it out /
3.    $X_i$=L($X_i$)              /see above Amendment 1/
4.    If $Y_i$.not.⊆$X_i$ Then
5.      $Y_i$=$X_i$∪$Y_i$      /restore non-redundant/
6.      L=L∪{$X_i$→$Y_i$}  /possibly non-full implications/
7.    Endif
8.  Endfor
9.  SORT L by lexicographic order on $X_i$(i∈I)
10. Basis=∅           /namely: 00<01<10<11.../
11. For i∈I
12.   For ($X_k$→$Y_k$)∈Basis    /is $X_i$ L-pseudo-closed?/
13.     If $X_k$⊆$X_i$.and.$X_i$.not.⊃$Y_k$ Then
14.        L=L\{$X_i$→$Y_i$}     /no: drop it out /
15.        Goto 20                  /i.e. Endfor i/
16.     Endif
17.   Endfor
18. $Y_i$=L($Y_i$)              /yes: make $X_i$→$Y_i$ full/
19. Basis=Basis∪{$X_i$→$Y_i$}
20 Endfor
```

Remarks. As compared with Alan Day's Algorithm 2, the only new idea is to separate saturating the premises although keeping enough information from the original implications (first loop), from detecting the L-pseudo-closed -by using their recursive characterization (Lemma 2.3, which requires sorting L before, statement 9)- and closing their conclusions (second loop). Even if this sorting and detection (statements 12-17) have a cost, a major benefit is to L-close *only* the L-pseudo-closed (statement 18), which will be more efficient when L comprises far more implications than L-pseudo-closed, by saving the deadly price of reiteration in calculating L(). This algorithm can also be done in place by permuting L to avoid the extra table "Basis".

Discussion.

Starting from an (a priori) non-full list of implications $L:=\{X_i \rightarrow Y_i \mid i \in I,\ X_i \subset Y_i \subseteq A\}$ for calculating its canonical basis $B_L:=\{X_k \rightarrow L(X_k) \mid k \in K,\ X_k$ L-pseudo-closed$\}$, any algorithm will so far require $|I|+|K|$ closures, unless a sparing-closure-test is elaborated in the future to detect non-L-pseudo-closed premises at first sight...

Algorithm 1 requires $|I|$ closures, but half the work is already embedded and was supposed done by the fullness hypothesis, to be honest. Notice that in this note it is the only procedure giving optionally at no memory cost a basis with the original smaller premises -that may be crucial in some applications either for semantical reasons (because the user care them) or for optimization (cost, minimal generators…). L should usually shrink quickly during the reduction process since -in addition to redundant implications- L-quasi-non-pseudo-closed are dropped out at first sight thanks to fullness, which so doing provides a non order-dependent algorithm.

The best feature of Alan Day's Algorithm 2 is to start with over-saturating premises that drops out redundant implications of which the premise L-closure is not essential, leading to $|I|+|K'|$ closures, $|K'| \in [|K|, |I|]$, but is order dependent.

Algorithm 3 is a variation which requires exactly the optimal $|I|+|K|$ closures, after a pre-sorting -instead of post-sorting as for Alan's-, with an extra $|I| \times$ loops in the growing list to detect L-pseudo-closed premises, replacing iterative closures by simple loops, which follows the popular advice: "better clean before you close"!

All three algorithms share two nice properties: they don't calculate L-saturation actually -which would involve reiteration and presuppose the L-closure at hand- but do a test leading "automatically" to pseudo / quasi-closed premises, and secondly the reduction processes can be done in place, which is nice for building general programs.

In that respect, since the beginning of the development of our computer program GLAD (General Lattice Analysis & Design, see [D83-96]), a main concern has not been so much a pernickety fight against complexity, but to try understanding the interplay between the representations of algebraic / structural objects in programs, methodological questions ([D99]), and a search for simple / clearer algorithms. Alan's algorithm is very elegant and doesn't require sophisticated (often valued although exponential…) constructions, even if Alan (or his hidden editor) was perhaps a bit unfair when claiming that another alternative algorithm [NextClosure for implications that starts from a "*formal*" *context*] "provides an excellent (though *necessarily exponential time*) algorithm" [Day92 p.410]. Unfair, because … the input are not the same! If one reduces something (a list) which is close to our current goal (a non-redundant list), one is for sure in a better situation than wandering in the exponential powerset of all potential premises even with clever shortcuts… In practice, both types of algorithms are absolutely necessary depending on the context and applications.

Remaining questions for the future will be: Are there other ideas than fullness / pre-sorting to optimize Alan Day's algorithm? Combining different approaches and tricks, will there be other properties for identifying pseudo-closed premises quickly? Is there any hidden algebraic property or tool that could drive their detection?

So far, the three algorithms that we focused on in this note give another discrete clue on why / how pseudo-closed premises are hard to catch: as the ultimate survivors among the original list of candidates, may be they are just "the veracious core of what is left after throwing away the unnecessary gangue", in an Hegelian spirit...

Acknowledgements
The writing of this note was postponed for many years due to a special emotion. Its first motivation came out of discussions with Alan Day, along with his friends Christian Herrmann, Douglas Pickering and Michael Roddy who brought me to Thunder Bay -so scared and feeling such an amateur- in April 1990, a few months before Alan's untimely death. After open talks, accompanying us to his front door, Alan told me with a grave smile "The only thing which is wrong is me: so, we don't meet again". I could not swallow and until today have not been able to utter a word.

Years afterwards, a first draft was at last prepared for the workshop "*Le treillis Rochelais*", La Rochelle, May 2007, for which many thanks are due to the organizers. Thanks are also due to the referees for their remarks aiming at clarifying this paper.

References

[B06] Bertet K., Some algorithmic aspects using the canonical direct implicational basis, in *CLA'06: Concept Lattices and their Applications* (S. Ben Yahia and E. Mephu Nguifo eds), Hammamet (2006) 101-114.

[Day92] Day A., The lattice theory of functional dependencies and Normal decompositions, *Intern. J. of Algebra and Computations* 2 (4) (1992) 409-431.

[D84-87] Duquenne V., Contextual implications between attributes and some representation properties for finite lattices. in *Beitrage zur Begriffsanalyse*, (B. Ganter, R. Wille and K.E. Wolf eds), 1987, Mannheim: 213-239.

[D83-96] Duquenne V., GLAD (General Lattice Analysis & Design): a program for a glad user, *ORDAL 96*: *Order and decision-making* (I. Rival ed.), Ottawa, www.csi.uottawa.ca.

[D99] Duquenne V., Latticial structures in Data Analysis, *ORDAL 96*: *Order and decision-making* (I. Rival ed.), Ottawa,www.csi.uottawa.ca, *Theo. Comp. Sci. 217* (1999) 407-436.

[D&Al01] Duquenne V., Chabert C., Cherfouh A, Delabar J.-M., Doyen A.-L, Pickering D., Structuration of phenotypes / genotypes through Galois lattices and implications, in *Proc. of ICCS2001-CLKDD'01* (Stanford 07/2001, E. Mephu Nguifo et *al.* eds) 21-32, and *Applied Artificial Intelligence 17* (2003) 243-256.

[G84-87] Ganter B., Algorithmen zur Formalen Begriffsanalyse in *Beitrage zur Begriffsanalyse*, (B. Ganter, R. Wille and K.E. Wolf ed.), 1987, Mannheim: 213-239 (preprint TH Darmstadt, 1984).

[GW99] Ganter B., Wille R., *Formal Concept Analysis, Mathematical Foundations.* Springer Verlag, Berlin, 1999.

[GD84-86] Guigues J.L., Duquenne.V., Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques & Sciences Humaines 95* (1986) 5-18, (preprint Groupe Mathématiques et Psychologie, Université Paris V, 1984).

[Ma83] Maier D., *The theory of relational databases*, Computer Science Press, 1983.

[O&D03-07] Obiedkov S., Duquenne V., Incremental construction of the canonical basis, in *JIM'2003* (M. Nadif, A. Napoli, E. SanJuan, A. Sigayret eds) 15-23, and accepted in *Annals of Mathematics and Artificial Intelligence 49* (2007) 77-99.

[R07] Rudolf S., Some notes on pseudo-closed sets, in ICFCA'07: Formal Concept Analysis (S.O. Kuznetsov and S. Schmidt eds), Clermont-Ferrand (2007), LNAI 4390, 151-165.

[V&D03-07] Valtchev P., Duquenne V. Towards scalable divide-and-conquer methods for computing concepts and implications, in *JIM'2003* (M. Nadif, A. Napoli, E. SanJuan, A. Sigayret eds) 2-12.

[W95] Wild M., Computations with finite closure systems and implications. In *Proceedings of the 1st Annual International Conference on Computing and Combinatorics,* volume 959 of *LNCS,* pages 111-120. Springer, 1995.

ZART: A Multifunctional Itemset Mining Algorithm

Laszlo Szathmary[1], Amedeo Napoli[1], and Sergei O. Kuznetsov[2]

[1] LORIA UMR 7503, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France
`{szathmar, napoli}@loria.fr`
[2] Higher School of Economics, Department of Applied Mathematics
Kirpichnaya 33/5, Moscow 105679, Russia
`skuznetsov@hse.ru`

Abstract. In this paper, we present and detail a multifunctional itemset mining algorithm called *Zart*, which is based on the *Pascal* algorithm. *Zart* shows a number of additional features and performs the following, usually independent, tasks: identify frequent closed itemsets and associate generators to their closures. This makes *Zart* a complete algorithm for computing classes of itemsets including generators and closed itemsets. These characteristics allow one to extract minimal non-redundant association rules, a useful and lossless representation of association rules. In addition, being based on the *Pascal* algorithm, *Zart* has a rather efficient behavior on weakly and strongly correlated data. Accordingly, *Zart* is at the heart of the Coron platform, which is a domain independent, multi-purposed data mining platform, incorporating a rich collection of data mining algorithms.

1 Introduction

Finding association rules is one of the most important tasks in data mining. Generating valid association rules from frequent itemsets (FIs) often results in a huge number of rules, which limits their usefulness in real life applications. To solve this problem, different concise representations of association rules have been proposed, e.g. generic basis ($\mathcal{GB}$) [1], informative basis ($\mathcal{IB}$) [1], representative rules [2], Duquennes-Guigues basis [3], Luxenburger basis [4], proper basis [5], structural basis [5], etc. A very good comparative study of these bases can be found in [6], where it is stated that a rule representation should be *lossless* (should enable derivation of all valid rules), *sound* (should forbid derivation of rules that are not valid), and *informative* (should allow determination of rules parameters such as support and confidence).

Kryszkiewicz showed in [6] that minimal non-redundant rules[3] ($\mathcal{MNR}$) with the cover operator, and the transitive reduction of minimal non-redundant rules[3] ($\mathcal{RMNR}$) with the cover operator and the confidence transitivity property are lossless, sound, and informative representations of *all* valid association rules.

[3] Defined in Section 2.

From the definitions of $\mathcal{MNR}$ and $\mathcal{RMNR}$ it can be seen that we only need frequent closed itemsets *and* their generators to produce these rules. Frequent itemsets have several condensed representations, e.g. closed itemsets [7–9], generator representation [10], approximate free-sets [11], disjunction-free sets [12], disjunction-free generators [10], generalized disjunction-free generators [13], non-derivable itemsets [14], etc. From these representations, the one which consists of frequent closed itemsets and frequent generators gives rise to a concise set of association rules, which is lossless, sound, and informative [6]. This set of rules, called the set of minimal non-redundant association rules ($\mathcal{MNR}$) [1], is not minimal in general case, but presents a good compromise between its size and time needed to generate it [15].

In [16], Bastide *et al.* presented the *Pascal* algorithm and claimed that $\mathcal{MNR}$ can be extracted with this algorithm. However, to obtain $\mathcal{MNR}$ from the output of *Pascal*, one has to do a lot of computing. First, frequent closed itemsets must also be known. Second, frequent generators must be *associated* to their closures. Here we propose an algorithm called *Zart*, an extension of *Pascal*, which does this computing. Thus, *Zart* allows one to easily construct $\mathcal{MNR}$. Instead of *Pascal*, we might have selected another algorithm. The reason for choosing *Pascal* was as follows: among *levelwise* frequent itemset mining algorithms, it may be the most efficient. This is due to its pattern counting inference mechanism that can significantly reduce the number of expensive database passes. Furthermore, as it was argued in [17], the idea introduced in *Zart* can be generalized, and thus it can be applied to *any* frequent itemset mining algorithm.

The paper is organized as follows. In the next section, we overview the basic concepts and essential definitions. This is followed by the description of the three main features of the *Zart* algorithm. We then present *Zart* and give a running example. Then, the generation of minimal non-redundant association rules is presented. Next, we provide experimental results for comparing the efficiency of *Zart* to *Pascal* and *Apriori*. Finally, we draw conclusions in the last section.

2 Main Definitions

Frequent Itemsets. We consider a set of *objects* $O = \{o_1, o_2, \dots, o_m\}$, a set of *attributes* $A = \{a_1, a_2, \dots, a_n\}$, and a binary relation $R \subseteq O \times A$, where $R(o, a)$ means that the object o has the attribute a. In formal concept analysis the triple (O, A, R) is called a *formal context* [18]. The Galois connection for (O, A, R) is defined along the lines of [18] in the following way (here $B \subseteq O$, $D \subseteq A$):

$$B' = \{a \in A \mid R(o, a) \text{ for all } o \in B\}, \qquad D' = \{o \in O \mid R(o, a) \text{ for all } a \in D\}.$$

In data mining applications, an element of A is called an *item* and a subset of A is called an *itemset*. Further on, we shall keep to these terms. An itemset of size i is called an i-itemset.[4] We say that an itemset $P \subseteq A$ *belongs* to an object

[4] For instance, {a,b,e} is a 3-itemset. Further on we use separator-free set notations, i.e. *abe* stands for {a,b,e}.

$o \in O$, if $(o, p) \in R$ for all $p \in P$, or $P \subseteq o'$. The *support* of an itemset $P \subseteq A$ indicates the number of objects to which the itemset belongs: $supp(P) = |P'|$. An itemset is *frequent* if its support is not less than a given *minimum support* (denoted by *min_supp*). An itemset P is *closed* if there exists no proper superset with the same support. The closure of an itemset P (denoted by P'') is the largest superset of P with the same support. Naturally, if $P = P''$, then P is a closed itemset. The task of frequent itemset mining consists of generating all (closed) itemsets (with their supports) with supports greater than or equal to a specified *min_supp*.

Two itemsets $P, Q \subseteq A$ are said to be *equivalent* ($P \cong Q$) iff they belong to the same set of objects (i.e. $P' = Q'$). The set of itemsets that are equivalent to an itemset P (P's *equivalence class*) is denoted by $[P] = \{Q \subseteq A \mid P \cong Q\}$. An itemset $P \in [P]$ is called a *generator*[5], if P has no proper subset in $[P]$, i.e. it has no proper subset with the same support. A *frequent generator* is a generator whose support is not less than a given minimum support.

Pattern Counting Inference. The following properties of support and generators were observed in [16] and are usually referred to as properties of *counting inference.*

Property 1. Let P and Q be two itemsets.
(i) $P \cong Q \Rightarrow supp(P) = supp(Q)$
(ii) $P \subseteq Q$ and $(supp(P) = supp(Q)) \Rightarrow P \cong Q$

Property 2. All subsets of a frequent generator are frequent generators.

Property 3. An itemset P is a generator iff $supp(P) \neq min_{p \in P}(supp(P \setminus \{p\}))$.

The last property says that in order to decide whether a candidate set P is a generator, one needs to compare its support to its subsets of size $|P| - 1$. By definition, generators do not admit proper subsets of the same support.

Frequent Association Rules. An association rule is an expression of the form $I_1 \rightarrow I_2$, where I_1 and I_2 are arbitrary itemsets ($I_1, I_2 \subseteq A$), $I_1 \cap I_2 = \emptyset$ and $I_2 \neq \emptyset$. The left side, I_1 is called *antecedent*, the right side, I_2 is called *consequent*. The support of an association rule[6] r is defined as: $supp(r) = supp(I_1 \cup I_2)$. The *confidence* of an association rule r: $I_1 \rightarrow I_2$ is defined as the conditional probability that an object has itemset I_2, given that it has itemset I_1: $conf(r) = supp(I_1 \cup I_2)/supp(I_1)$. An association rule r with $conf(r) = 100\%$ is an *exact* association rule (or implication [18]), otherwise it is an *approximate* association rule. An association rule is *valid* if $supp(r) \geq min\_supp$ and $conf(r) \geq min\_conf$. The set of valid association rules is denoted by $\mathcal{AR}$.

Now recall the following definitions of bases of association rules:

[5] In the literature these itemsets have various names: key itemsets, minimal generators, free-itemsets, key generators, etc. Further on we will refer to them as "generators".
[6] In this paper we use absolute values, but the support of an association rule r is also often defined as $supp(r) = supp(I_1 \cup I_2)/|O|$.

Definition 1. *Let FCI be the set of frequent closed itemsets. For each frequent closed itemset f, let FG_f denote the set of frequent generators in the equivalence class of f. The generic basis for exact association rules (implications):*

$$\mathcal{GB} = \{r : g \Rightarrow (f \backslash g) \mid f \in FCI \wedge g \in FG_f \wedge g \neq f\}.$$

Definition 2. *Let FCI be the set of frequent closed itemsets and let FG be the set of frequent generators. The informative basis for approximate association rules:*

$$\mathcal{IB} = \{r : g \rightarrow (f \backslash g) \mid f \in FCI \wedge g \in FG \wedge g'' \subset f\}.$$

Definition 3. *Let $\mathcal{IB}$ denote the informative basis for approximate association rules as defined above, and let FCI be the set of frequent closed itemsets. The transitive reduction of $\mathcal{IB}$:*

$$\mathcal{RIB} = \{r : g \rightarrow (f \backslash g) \in \mathcal{IB} \mid g'' \textit{ is a maximal proper subset of } f \textit{ in } FCI\}.$$

Definition 4. *The set of minimal non-redundant rules ($\mathcal{MNR}$) is defined as: $\mathcal{MNR} = \mathcal{GB} \cup \mathcal{IB}$. The transitive reduction of minimal non-redundant rules ($\mathcal{RMNR}$) is defined as: $\mathcal{RMNR} = \mathcal{GB} \cup \mathcal{RIB}$.*

3 Main Features of Zart

Zart has three main features, namely **(1)** pattern counting inference, **(2)** identifying frequent closed itemsets, and **(3)** identifying generators of frequent closed itemsets.

3.1 Pattern Counting Inference in Pascal and Zart

The first part of *Zart* is based on *Pascal*, which employs properties of the counting inference. In levelwise traversal of frequent itemsets, first the smallest elements of an equivalence class are discovered, and these itemsets are exactly the generators. Later, when finding a larger itemset, it is tested if it belongs to an already discovered equivalence class. If it does, the database does not have to be accessed to determine the support of the itemset. This way the expensive database passes and support counts can be constrained to the case of generators only. From some level on, all generators can be found, thus all remaining frequent itemsets and their supports can be inferred without any further database pass.

In Figure 1 (left) we show the output of *Pascal* when executed on dataset $\mathcal{D}$ (Table 4): it finds frequent itemsets and marks frequent generators. Recalling the definitions of $\mathcal{MNR}$ and $\mathcal{RMNR}$, we see that this output is not enough. From our running example, the output of *Zart* is shown in Figure 1 (right). Here one can see the equivalence classes of database $\mathcal{D}$. Only the maximal (frequent closed itemset) and minimal elements (frequent generators) of each equivalence class are indicated. Support values are shown in the top right-hand corner of classes. As can be seen, the output of *Zart* is necessary and sufficient for generating $\mathcal{GB}$, $\mathcal{IB}$, $\mathcal{RIB}$, $\mathcal{MNR}$, and $\mathcal{RMNR}$.

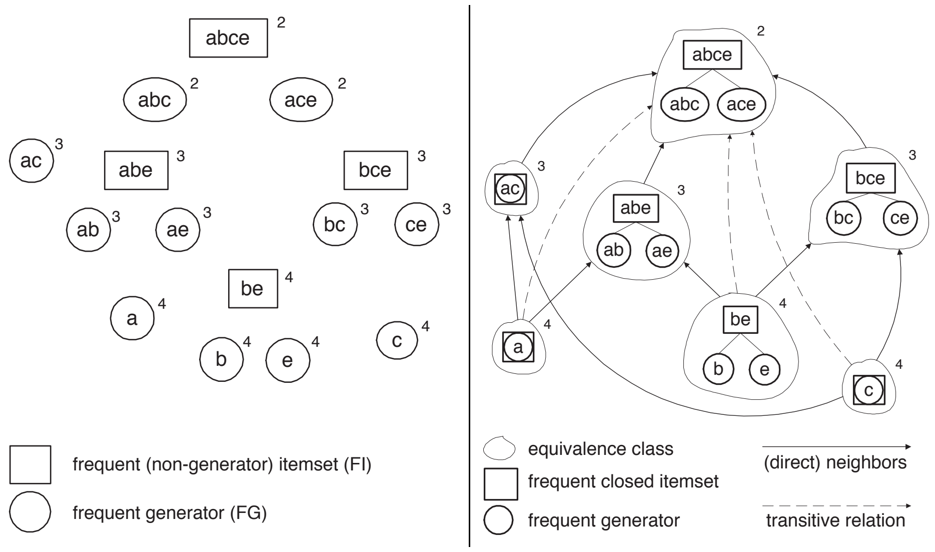


Fig. 1. Result of *Pascal* (left) and *Zart* (right) on $\mathcal{D}$ with $min\_supp = 2$ (40%)

3.2 Identifying Closed Itemsets among Frequent Itemsets in Zart

The second part of *Zart* consists in the identification of FCIs among FIs, adapting this idea from *Apriori-Close* [5]. By definition, a closed itemset has no proper superset with the same support. At the i^{th} step all i-itemsets are marked as "closed". At the $(i+1)^{th}$ iteration for each $(i+1)$-itemset we test if it contains an i-itemset with the same support. If so, then the i-itemset is not a closed itemset since it has a proper superset with the same support, thus it is marked as "not closed". When the algorithm terminates with the enumeration of all FIs, itemsets still marked "closed" are the FCIs of the dataset.

3.3 Associating the Generators to their Closures in Zart

Because of the levelwise itemset search, when an FCI is found, all its frequent subsets are already known. This means that its generators are already computed, they only have to be identified. We show that the search space for generators can be narrowed to not closed ones. This is justified by the following properties:

Property 4. A closed itemset cannot be a generator of a larger itemset.

Property 5. The closure of a frequent not closed generator g is the smallest proper superset of g in the set of frequent closed itemsets.

By using these two properties, the algorithm for efficiently finding generators is the following: generators are stored in a list l. At the i^{th} iteration, frequent closed i-itemsets are filtered. For each frequent closed i-itemset z, the following steps are executed: find the subsets of z in list l, register them as generators of z, and delete them from l. Before passing to the $(i{+}1)^{th}$ iteration, add the i-itemsets that are not closed generators to list l. Properties 4 and 5 *guarantee* that whenever the subsets of a frequent closed itemset are looked for in list l, only its generators are returned. The returned subsets have the same support as the frequent closed itemset, it does not even have to be tested. Since only the generators are stored in the list, it means that we need to test far fewer elements than the whole set of FIs. Since at step i the size of the largest itemset in list l can be maximum $(i - 1)$, we do not find the generators that are identical to their closures. If an FCI has no generator registered, it simply means that its generator is itself. As for the implementation, instead of using a "normal" list for storing generators, the trie data structure is suggested, since it allows a very quick lookup of subsets.

4 The Zart Algorithm

4.1 Pseudo Code

Due to lack of space, we can only give a short overview of the algorithm here. The detailed description of *Zart* can be found in [19]. The main block of the algorithm is given in Table 2. *Zart* uses three different kinds of tables, their descriptions are provided in Table 1. We assume that an itemset is an ordered list of attributes, since we will rely on this in the Zart-Gen function.

Table 1. Tables and table fields used in *Zart*

C_i	potentially frequent candidate i-itemsets fields: (1) itemset, (2) pred_supp, (3) key, (4) support
F_i	frequent i-itemsets fields: (1) itemset, (2) key, (3) support, (4) closed
Z_i	frequent closed i-itemsets fields: (1) itemset, (2) support, (3) gen

itemset	– an arbitrary itemset
pred_supp	– the minimum of the supports of all $(i-1)$-long frequent subsets
key	– is the itemset a generator?
closed	– is the itemset a closed itemset?
gen	– generators of a closed itemset

Zart-Gen function. As input, it gets an F_i table that has a set of frequent itemsets. As output, it returns the C_{i+1} table. The method is the following. It fills the C_{i+1} table with the one-size larger supersets of the itemsets in F_i. The pred_supp values in C_{i+1} are set to the minimum of the supports of all one-size smaller subsets. If a subset is not a generator, then the current itemset is not a generator either, and thus its support is equal to its pred_supp value.

Find-Generators procedure. As input it gets a Z_i table. The method is the following. For each frequent closed itemset z in Z_i, find its proper subsets in the global FG list, register them as generators of z, delete them from FG, and add not closed generators from F_i to FG.

Table 2. Main block of *Zart*

```
 1) $FG \leftarrow \{\ \}$; // global list of frequent generators
 2) fill $C_1$ with 1-itemsets and count their supports;
 3) copy frequent itemsets from $C_1$ to $F_1$;
 4) mark itemsets in $F_1$ as "closed";
 5) mark itemsets in $F_1$ as "key" if their support $< |O|$;
       //where $|O|$ is the number of objects in the input dataset
 6) if there is a full column in the input dataset, then $FG \leftarrow \{\emptyset\}$;
 7) $i \leftarrow 1$;
 8) loop
 9) {
10)       $C_{i+1} \leftarrow$ Zart-Gen($F_i$);
11)       if $C_{i+1}$ is empty then break from loop;
12)       count the support of "key" itemsets in $C_{i+1}$;
13)       if $C_{i+1}$ has an itemset whose support = pred_supp,
              then mark it as "not key";
14)       copy frequent itemsets to $F_{i+1}$;
15)       if an itemset in $F_{i+1}$ has a subset in $F_i$ with the same
              support, then mark the subset as "not closed";
16)       copy "closed" itemsets from $F_i$ to $Z_i$;
17)       Find-Generators($Z_i$);
18)       $i \leftarrow i + 1$;
19) }
20) copy itemsets from $F_i$ to $Z_i$;
21) Find-Generators($Z_i$);
```

4.2 Running Example

The execution of *Zart* on dataset $\mathcal{D}$ (Table 4, left) is illustrated in Table 3. The algorithm first performs one database scan to count the support values of 1-itemsets. The itemset d is pruned because it is not frequent. At the next iteration, all candidate 2-itemsets are created and their support values are counted. In C_2 there is one itemset with the same support as one of its subsets, thus be is not a generator. Using F_2, the itemsets b and e in F_1 are not closed because they have a proper superset with the same support. The remaining closed itemsets a and c are copied to Z_1 and their generators are determined. In the global list of frequent generators (FG), which is still empty, they have no subsets, which means that both a and c are generators themselves. Not closed generators of F_1 (b and e) are added to the FG list. In C_3, abe and bce turn out to be not generators. Their support values are equal to the support of be (Property 3). By F_3, the itemsets ab, ae, bc, and ce turn out to be not closed. The remaining closed itemsets ac and be are copied to Z_2. The generator of ac is itself, and the generators of be are b and e. These two generators are deleted from FG and ab, ae, bc, and ce are added to FG. The candidate $abce$ is not a generator either, and as there are no more candidate generators in C_4, from this step on no more database scan is needed. In the fifth iteration no new candidate is found and the algorithm breaks out from the main loop. The generators of $abce$ are read from FG. When

Table 3. Execution of *Zart* on dataset $\mathcal{D}$ with $min\_supp = 2$ (40%)

DB scan$_1$ $\rightarrow$

C_1	pred_supp	key	supp
a			4
b			4
c			4
d			1
e			4

F_1	key	supp	closed
a	yes	4	yes
b	yes	4	~~yes~~
c	yes	4	yes
e	yes	4	~~yes~~

Z_1	supp	gen
a	4	
c	4	

$FG_{before} = \{\}$
$FG_{after} = \{b, e\}$

DB scan$_2$ $\rightarrow$

C_2	pred_supp	key	supp
ab	4	yes	3
ac	4	yes	3
ae	4	yes	3
bc	4	yes	3
be	4	~~yes~~	4
ce	4	yes	3

F_2	key	supp	closed
ab	yes	3	~~yes~~
ac	yes	3	yes
ae	yes	3	~~yes~~
bc	yes	3	~~yes~~
be	no	4	yes
ce	yes	3	~~yes~~

Z_2	supp	gen
ac	3	
be	4	$\{b, e\}$

$FG_{before} = \{b, e\}$
$FG_{after} = \{ab, ae, bc, ce\}$

DB scan$_3$ $\rightarrow$

C_3	pred_supp	key	supp
abc	3	yes	2
abe	3	~~yes~~	3
ace	3	yes	2
bce	3	~~yes~~	3

F_3	key	supp	closed
abc	yes	2	~~yes~~
abe	no	3	yes
ace	yes	2	~~yes~~
bce	no	3	yes

Z_3	supp	gen
abe	3	$\{ab, ae\}$
bce	3	$\{bc, ce\}$

$FG_{before} = \{ab, ae, bc, ce\}$
$FG_{after} = \{abc, ace\}$

C_4	pred_supp	key	supp
$abce$	2	~~yes~~	2

F_4	key	supp	closed
$abce$	no	2	yes

Z_4	supp	gen
$abce$	2	$\{abc, ace\}$

$FG_{before} = \{abc, ace\}$
$FG_{after} = \{\}$

C_5	pred_supp	key	supp
$\emptyset$			

the algorithm stops, all FIs and all FCIs *with* their generators are determined, as shown in Table 4 (right). In the table the "+" sign means that the frequent itemset is closed. The support values are indicated in parentheses. If *Zart* leaves the generators of a closed itemset empty, it means that the generator is identical to the closed itemset (as this is the case for a, c, and ac in the example).

5 Finding Minimal Non-Redundant Association Rules with Zart

Generating all valid association rules from FIs produces too many rules of which many are redundant. For instance, in dataset $\mathcal{D}$ with $min\_supp = 2$ (40%) and $min\_conf = 50\%$ no less than 50 rules can be extracted. Considering the small size of the dataset, 5×5, this quantity is huge. How could we find the most interesting rules? How could we avoid redundancy and reduce the number of rules? Minimal non-redundant association rules ($\mathcal{MNR}$) can help us. By Definition 4, an $\mathcal{MNR}$ has the following form: the antecedent is an FG, the

Table 4. A sample dataset $\mathcal{D}$ for the examples (left) and the output of *Zart* (right)

	a	b	c	d	e
1	x	x		x	x
2	x		x		
3	x	x	x		x
4		x	x		x
5	x	x	x		x

All frequent itemsets ($\bigcup_i F_i$)		All frequent closed itemsets with their generators ($\bigcup_i Z_i$)
a (4) +	*be* (4) +	*a* (4); [*a*]
b (4)	*ce* (3)	*c* (4); [*c*]
c (4) +	*abc* (2)	*ac* (3); [*ac*]
e (4)	*abe* (3) +	*be* (4); [*b*, *e*]
ab (3)	*ace* (2)	*abe* (3); [*ab*, *ae*]
ac (3) +	*bce* (3) +	*bce* (3); [*bc*, *ce*]
ae (3)	*abce* (2) +	*abce* (2); [*abc*, *ace*]
bc (3)		

Table 5. Comparing sizes of different sets of association rules

dataset (min_supp)	min_conf	$\mathcal{AR}$ (all valid rules)	$\mathcal{GB}$	$\mathcal{IB}$	$\mathcal{RIB}$	$\mathcal{MNR}$ ($\mathcal{GB} \cup \mathcal{IB}$)	$\mathcal{RMNR}$ ($\mathcal{GB} \cup \mathcal{RIB}$)
$\mathcal{D}$ (40%)	50%	50	8	17	13	25	21
T20I6D100K (0.5%)	90%	752,715	232	721,716	91,422	721,948	91,654
	70%	986,058		951,340	98,097	951,572	98,329
	50%	1,076,555		1,039,343	101,360	1,039,575	101,592
	30%	1,107,258		1,068,371	102,980	1,068,603	103,212
C20D10K (30%)	90%	140,651	967	8,254	2,784	9,221	3,751
	70%	248,105		18,899	3,682	19,866	4,649
	50%	297,741		24,558	3,789	25,525	4,756
	30%	386,252		30,808	4,073	31,775	5,040
C73D10K (90%)	95%	1,606,726	1,368	30,840	5,674	32,208	7,042
	90%	2,053,936		42,234	5,711	43,602	7,079
	85%	2,053,936		42,234	5,711	43,602	7,079
	80%	2,053,936		42,234	5,711	43,602	7,079
MUSHROOMS (30%)	90%	20,453	544	952	682	1,496	1,226
	70%	45,147		2,961	1,221	3,505	1,765
	50%	64,179		4,682	1,481	5,226	2,025
	30%	78,888		6,571	1,578	7,115	2,122

union of the antecedent and consequent is an FCI, and the antecedent is a proper subset of this FCI. For the generation of such rules, the FCIs *and* their associated generators are needed. Since *Zart* can find both, the output of *Zart* (Table 4, right) can be used directly to generate these rules. For a very quick lookup of frequent closed proper supersets of generators we suggest storing FCIs in a trie.

The algorithm for finding $\mathcal{MNR}$ is the following: for each FG P_1 find its proper supersets P_2 in the set of FCIs. Then add the rule $r : P_1 \rightarrow P_2 \setminus P_1$ to the set of $\mathcal{MNR}$. For instance, using the generator e in Figure 1 (right), three rules can be determined. Rules within an equivalence class form the generic basis ($\mathcal{GB}$), which consists of exact association rules ($e \Rightarrow b$), while rules between equivalence classes are approximate association rules ($e \rightarrow bc$ and $e \rightarrow abc$). For extracting $\mathcal{RMNR}$, the search space for finding frequent closed proper supersets of generators is reduced to equivalence classes that are *direct neighbors*, i.e. transitive relations are eliminated. Thus, for instance, in the previous example only the first two rules are generated: $e \Rightarrow b$ and $e \rightarrow bc$. A comparative table of the different sets of association rules, that can be extracted easily using the output of *Zart*, is shown in Table 5.[7] In sparse datasets, like T20I6D100K, the number of $\mathcal{MNR}$ is not much less than the number of $\mathcal{AR}$. However, in dense, highly correlated datasets, like C20D10K or Mushrooms, the difference is significant. $\mathcal{RMNR}$ always represents much less rules than $\mathcal{AR}$, in sparse and dense datasets too.

6 Experimental Results

We evaluated *Zart* against *Apriori* and *Pascal*. We have implemented these algorithms in Java using the same data structures, and they are all part of the Coron data mining platform [20]. The experiments were carried out on an Intel Pentium IV 2.4 GHz machine running Debian GNU/Linux operating system with 512 MB RAM. All times reported are real, wall clock times as obtained from the Unix *time* command between input and output. For the experiments we have used the following datasets: T20I6D100K, C20D10K, and Mushrooms. The T20I6D100K[8] is a sparse dataset, constructed according to the properties of market basket data that are typical weakly correlated data. The C20D10K is a census dataset from the PUMS sample file, while the Mushrooms[9] describes mushrooms characteristics. The last two are highly correlated datasets. It has been shown that weakly correlated data, such as synthetic data, constitute easy cases for the algorithms that extract frequent itemsets, since few itemsets are frequent. For such data, all algorithms give similar response times. On the contrary, dense and highly-correlated data constitute far more difficult cases for the extraction due to large differences between the number of frequent and frequent closed itemsets. Such data represent a huge part of real-life datasets. Response times for the datasets are presented numerically in Table 6.

6.1 Weakly Correlated Data

The T20I6D100K synthetic dataset mimics market basket data that are typical sparse, weakly correlated data. In this dataset, the number of FIs is small and nearly all FIs are generators. *Apriori*, *Pascal*, and *Zart* behave identically. As we can see in T20I6D100K, above 0.75% minimum support all frequent itemsets are closed and generators at the same time. It means that each equivalence class has one element only. Because of this, *Zart* and *Pascal* cannot use the advantage of pattern counting inference and they work exactly like *Apriori*.

6.2 Strongly Correlated Data

In datasets C20D10K and Mushrooms, the number of FGs is much less than the total number of FIs. Hence, using pattern counting inference, *Zart* has to

[7] Note that in the case of $\mathcal{GB}$, by definition, minimum confidence is 100%.
[8] http://www.almaden.ibm.com/software/quest/Resources/
[9] http://kdd.ics.uci.edu/

Table 6. Response times of *Zart* and other statistics (number of FIs, number of FCIs, number of FGs, proportion of the number of FCIs to the number of FIs, proportion of the number of FGs to the number of FIs)

min_supp	execution time (sec.)			# FIs	# FCIs	# FGs	$\frac{\#FCIs}{\#FIs}$	$\frac{\#FGs}{\#FIs}$
	Apriori	Pascal	Zart					
T20I6D100K								
2%	72.67	71.15	71.16	378	378	378	100.00%	100.00%
1%	107.63	106.24	107.69	1,534	1,534	1,534	100.00%	100.00%
0.75%	134.49	132.00	133.00	4,710	4,710	4,710	100.00%	100.00%
0.5%	236.10	228.37	230.17	26,836	26,208	26,305	97.66%	98.02%
0.25%	581.11	562.47	577.69	155,163	149,217	149,447	96.17%	96.32%
C20D10K								
50%	61.18	16.68	17.94	1,823	456	456	25.01%	25.01%
40%	71.60	19.10	19.22	2,175	544	544	25.01%	25.01%
30%	123.57	26.74	26.88	5,319	951	967	17.88%	18.18%
20%	334.87	53.28	54.13	20,239	2,519	2,671	12.45%	13.20%
10%	844.44	110.78	118.09	89,883	8,777	9,331	9.76%	10.38%
Mushrooms								
60%	3.10	2.04	2.05	51	19	21	37.25%	41.18%
50%	6.03	3.13	3.13	163	45	53	27.61%	32.52%
40%	13.93	6.00	6.03	505	124	153	24.55%	30.30%
30%	46.18	12.79	12.84	2,587	425	544	16.43%	21.03%
20%	554.95	30.30	34.88	53,337	1,169	1,704	2.19%	3.19%

perform much fewer support counts than *Apriori*. We can observe in all cases that the execution times of *Zart* and *Pascal* are almost identical: adding the FCI derivation *and* the identification of their generators to the FI discovery does not induce serious additional computation time. *Apriori* is very efficient on sparse datasets, but on strongly correlated data the other two algorithms perform much better.

7 Conclusion and Future Work

In this paper we presented a multifunctional itemset mining algorithm called *Zart*, which is a refinement of *Pascal*. As an addition, it can identify frequent closed itemsets, and it can associate generators to their closure. We showed that these extra features are essential for the generation of minimal non-redundant association rules. Experimental results show that *Zart* gives almost equivalent response times to *Pascal* on both weakly and strongly correlated data.

An interesting question is the following: can the idea of *Zart* be generalized and used for *any* arbitrary frequent itemset mining algorithm, be it either breadth-first or depth-first? Could we somehow extend these algorithms in a universal way to produce such results that can be used directly to generate not only all valid association rules, but minimal non-redundant association rules too? Our answer is positive [17], but detailed study of this will be subject of a next paper.

References

1. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining minimal non-redundant association rules using frequent closed itemsets. In Lloyd, J.*et al.*., ed.: Proc. of CL'00. Volume 1861 of LNAI., Springer (2000) 972–986
2. Kryszkiewicz, M.: Representative association rules. In: Proc. of PAKDD '98, London, UK, Springer-Verlag (1998) 198–209
3. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informatives résultant d'un tableau de données binaires. Math. et Sci. Hum. **95** (1986) 5–18
4. Luxenburger, M.: Implications partielles dans un contexte. Mathématiques, Informatique et Sciences Humaines **113** (1991) 35–55
5. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Closed set based discovery of small covers for association rules. In: Proc. of BDA '99. (1999) 361–381
6. Kryszkiewicz, M.: Concise Representations of Association Rules. In: Proc. of the ESF Exploratory Workshop on Pattern Detection and Discovery. (2002) 92–109
7. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient mining of association rules using closed itemset lattices. Inf. Syst. **24**(1) (1999) 25–46
8. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. LNCS **1540** (1999) 398–416
9. Zaki, M.J., Hsiao, C.J.: CHARM: An Efficient Algorithm for Closed Itemset Mining. In: Proc. of SDM '02. (2002) 33–43
10. Kryszkiewicz, M.: Concise representation of frequent patterns based on disjunction-free generators. In: Proc. of ICDM '01, Washington, DC, USA, IEEE Computer Society (2001) 305–312
11. Boulicaut, J.F., Bykowski, A., Rigotti, C.: Approximation of frequency queries by means of free-sets. In: Proc. of PKDD '00, Lyon, France, Springer Berlin / Heidelberg (2000) 75–85
12. Bykowski, A., Rigotti, C.: A condensed representation to find frequent patterns. In: Proc. of PODS '01, ACM Press (2001) 267–273
13. Kryszkiewicz, M., Gajek, M.: Why to apply generalized disjunction-free generators representation of frequent patterns? In: Proc. of ISMIS 2002, Lyon, France, Springer-Verlag Berlin / Heidelberg (2002) 383–392
14. Calders, T., Goethals, B.: Mining all non-derivable frequent itemsets. In: Proc. of PKDD '02, London, UK, Springer-Verlag (2002) 74–85
15. Pasquier, N.: Mining association rules using formal concept analysis. In: Proc. of ICCS '00, Shaker-Verlag (2000) 259–264
16. Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. SIGKDD Explor. Newsl. **2**(2) (2000) 66–75
17. Szathmary, L.: Symbolic Data Mining Methods with the Coron Platform (Méthodes symboliques de fouille de données avec la plate-forme Coron). PhD in Computer Sciences, Univ. Henri Poincaré Nancy 1, France (2006)
18. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer, Berlin/Heidelberg (1999)
19. Szathmary, L., Napoli, A., Kuznetsov, S.O.: ZART: A Multifunctional Itemset Mining Algorithm. LORIA Research Report 00001271 (2005)
20. Szathmary, L., Napoli, A.: CORON: A Framework for Levelwise Itemset Mining Algorithms. In: Suppl. Proc. of ICFCA '05, Lens, France. (2005) 110–113

Extensions of Bordat's algorithm for attributes

K. Bertet, S. Guillas, and J.-M. Ogier

Laboratory L3I - University of La Rochelle - France

Abstract. In our works, we use a concept lattice as classifier of noised symbol images. On the contrary of others methods of classification based on Formal Concept Analysis [12], our approach is adapted to the special case of noisy since it is based on a navigation into the lattice structure to classify a noised symbol image : the navigation is performed from the minimal concept, until a final concept is reached, according to the cover-relation between concepts. Class of the input noised symbol is then the class associated to the reached final concept.

We use Bordat's algorithm to generate the concept lattice since it generates the cover relation of the lattice. In this paper, we present three extensions of Bordat's algorithm : the first extension generates the reduction of the concept lattice to its attributes, i.e. a closure system on attributes ; the second extension generates concepts only when required during the navigation, thus a reduction of the total number of generated concepts ; the third extension generates the concept lattice together with the canonical direct basis, i.e. a basis of implication rules between attributes to describe them.

Key words: generation algoritm, concept lattice, classification, noised images

1 Introduction

In our current works in [16], we use a concept lattice as classifier of noised symbol images, where objects are symbols and attributes are features issued from images of symbols. On the contrary of others methods of classification based on Formal Concept Analysis [12], our approach is based on a navigation into the lattice structure to classify a symbol image : the navigation consists in a breadth-first search in the concept lattice starting from the bottom, until a final concept is reached, using a choice criteria to choose according to the cover-relation between concepts. Class of the input noised symbol is then the class associated to the reached final concept. This approach is similar to the use of a decision tree, adapted to the special case of noisy objects.

Bordat's algorithm is an appropriate algorithm to generate the concept lattice since it generates the cover relation between concepts. A concept is a pair (A, B) where A is a subset of attributes and B is a subset of objects. Since we only need attributes during the navigation in the concept lattice, Bordat's algorithm is first extended to compute the reduction of the concept lattice to the attributes. Such a reduction forms a closure system on attributes.

Only a small part of the concept lattice is explored by navigation, the others concepts are not required. Thus, the navigation implies the possibility to on-line generate a concept only when it is reached by navigation, aims of our second extension of Bordat's algorithm. As the main drawback of the concept lattice is its exponential complexity in the worst case, we understand the interest to build only the concepts necessary to the recognition.

Implication rules between attributes represents an efficient tools to describe relationship between attributes in different areas research (databases area [18] ; data-mining [26], [25] ...). Among equivalent implicational systems, there exists two basis : the canonical basis [14] that has the canonical and minimality properties ; the canonical direct basis [17] that has the directness, canonical and minimality properties. As a third extension of Bordat's algorithm, we describe the generation the canonical direct basis while generated of the cover relation of the concept lattice.

Section 2 gives definitions of a concept lattice, a closure system and an implicational system. The three extensions of Bordat's algorithm for classification of noisy symbols are descried in Section 3.

2 Definitions

In this paper, all sets are finite sets and a subset $X = \{x_1, x_2, \ldots, x_n\}$ is written as the word $X = x_1x_2\ldots x_n$. Moreover, we abuse notation and use $X + x$ (respectively, $X - x$) for $X \cup \{x\}$ (respectively, $X\backslash\{x\}$), with $X \subseteq S$ and $x \in S$.

Binary relation and lattice. An order relation $\leq$ on a set S is a binary relation on S which is reflexive ($\forall x \in S, x \leq x$), antisymmetric ($\forall x \neq y \in S$, $x \leq y$ imply $y \not\leq x$) and transitive ($\forall x, y, z \in S$, $x \leq y$ and $y \leq z$ imply $x \leq z$). A *partially ordered set* $P = (S, \leq)$, also called a *poset*, is a set S equipped with an order relation $\leq$. The cover relation $\prec$ on S is the transitive reduction of the order relation $\leq$. A poset $L = (S, \leq)$ is a *lattice* if any pair $\{x, y\}$ of elements of L has a *join* (i.e. a least upper bound) denoted by $x \vee y$ and a *meet* (i.e. a greatest lower bound) denoted by $x \wedge y$. Therefore, a lattice contains a minimum (resp. maximum) element according to the relation $\leq$ called the *bottom* (resp. *top*) of the lattice, and denoted $\bot_S$ or simply $\bot$ (resp. $\top_S$ or simply $\top$.)

An equivalence relation $\sim$ on a set S is a binary relation on S which is reflexive ($\forall x \in S, x \sim x$), symmetric ($\forall x \neq y \in S$, $x \sim y$ imply $y \sim x$) and transitive ($\forall x, y, z \in S$, $x \sim y$ and $y \sim z$ imply $x \sim z$). The equivalence class of $a \in S$ is the subset of all elements in S which are equivalent to a : $\lfloor a \rfloor = \{x \in S : x \sim a\}$. The set of all equivalence classes in S given an equivalence relation $\sim$ is usually denoted as $S/_\sim$, and called the quotient set of S by $\sim$. The quotient set $S/_\sim$ forms a partition of S.

Concept lattice. A *formal context* $K = (G, M, I)$ consists of two sets G and M, and a relation I between G and M. The elements of G are called the *objects*, and the elements of M are called the *attributes* of the context. We define the application f that associates to every element $o \in G$ the set $f(o) = \{a \in M \mid (o, a) \in I\}$, and the application g that associates to every element $a \in M$

the set $g(a) = \{o \in G \mid (o,a) \in I\}$. The extension of f to subsets $A \subseteq G$ provides :

$$f(A) = \cap_{o \in A} f(o) = \{a \in M \mid \forall\ o \in A,\ (o,a) \in I\ \} \tag{1}$$

Dually, the extension of g to subsets $B \subseteq M$ provides :

$$g(B) = \cap_{a \in G} g(a) = \{o \in G \mid \forall\ a \in B,\ (o,a) \in I\} \tag{2}$$

The two applications f and g forms a *Galois connection* between G and M.

A *formal concept* of a formal context K is a pair (A,B) with $A \subseteq G$, $B \subseteq M$, $f(A) = B$ and $g(B) = A$. Let $\mathbb{C}$ be the set of all the concepts of K, and $\leq_C$ be the following order relation on $\mathbb{C}$, with (A_1,B_1) and (A_2,B_2) two concepts :

$$(A_1,B_1) \leq_C (A_2,B_2) \text{ iff } A_1 \subseteq A_2 \text{ (or equivalently } B_2 \subseteq B_1) \tag{3}$$

The poset $(\mathbb{C}, \leq_C)$ is a lattice called the *concept lattice*. This lattice is also denoted *Galois lattice* by reference to the Galois connection (f,g) of the formal context C. In particular $\bot = (f(G), G)$ and $\top = (M, g(M))$. Moreover for all (A_1,B_1), $(A_2,B_2) \in \mathbb{C}$, $(A_1,B_1) \vee (A_2,B_2) = (f(B_1 \cap B_2), B_1 \cap B_2)$ and $(A_1,B_1) \wedge (A_2,B_2) = (A_1 \cap A_2, g(A_1 \cap A_2))$. In *formal concept analysis* (FCA) concept lattices are used to analyze data when organized by a binary relation between object and attributes. See the complete book of Ganter and Wille [12] for a complete description of formal concept analysis.

Closure system. A *set system* on a set S is a family of subsets of S. A *closure system* $\mathbb{F}$ on a set S, also called a *Moore family*, is a set system stable by intersection and which contains S : $S \in \mathbb{F}$ and $F_1, F_2 \in \mathbb{F}$ implies $F_1 \cap F_2 \in \mathbb{F}$. The subsets belonging to a closure system $\mathbb{F}$ are called the *closed sets* of $\mathbb{F}$.

The poset $(\mathbb{F}, \subseteq)$ is a lattice with, for each $F_1, F_2 \in \mathbb{F}$, $F_1 \wedge F_2 = F_1 \cap F_2$ and $F_1 \vee F_2 = \bigcap\{F \in \mathbb{F} \mid F_1 \cup F_2 \subseteq F\}$. Moreover, any lattice L is isomorphic to the lattice of closed sets of a closure system (see [2] for more details).

A *closure operator* on a set S is a map φ on $\mathcal{P}(S)$ satisfying the three following properties : φ is *isotone* (i.e. $\forall X, X' \subseteq S$, $X \subseteq X' \Rightarrow \varphi(X) \subseteq \varphi(X')$), *extensive* (i.e. $\forall X \subseteq S$, $X \subseteq \varphi(X)$) and *idempotent* (i.e. $\forall X \subseteq S$, $\varphi^2(X) = \varphi(X)$). Closure operators are in one-to-one mapping with closure systems. On the first hand, the set of all closed elements of φ forms a closure system $\mathbb{F}_\varphi$:

$$\mathbb{F}_\varphi = \{F \subseteq S \mid F = \varphi(F)\} \tag{4}$$

Dually, given a closure system $\mathbb{F}$ on a set S, one defines the closure $\varphi_\mathbb{F}(X)$ of a subset X of S as the least element $F \in \mathbb{F}$ that contains X :

$$\varphi_\mathbb{F}(X) = \bigcap\{F \in \mathbb{F} \mid X \subseteq F\} \tag{5}$$

In particular $\varphi_\mathbb{F}(\emptyset) = \bot_\mathbb{F}$. Moreover for all $F_1, F_2 \in \mathbb{F}$, $F_1 \vee F_2 = \varphi_\mathbb{F}(F_1 \cup F_2)$ and $F_1 \wedge F_2 = \varphi_\mathbb{F}(F_1 \cap F_2) = F_1 \cap F_2$.

A concept lattice $(\mathbb{C}, \leq_C)$ is composed of two particular closure systems $\mathbb{C}_G$ and $\mathbb{C}_M$ respectively defined on the set G of objects, and on the set M of attributes : $\mathbb{C}_G$ is the restriction of $\mathbb{C}$ to the objects where each concept (A, B) is replaced by the subset A of objects ; the associated closure operator is $f \circ g$. Dually, $\mathbb{C}_M$ is the restriction of $\mathbb{C}$ to the attributes and the associated closure operator is $g \circ f$. Moreover, the three lattices $(\mathbb{C}_G, \subseteq)$, $(\mathbb{C}_M, \supseteq)$ and $(\mathbb{C}, \leq)$ are isomorphic. See the survey of Caspard and Monjardet [5] for more details about closure systems.

Implicational system. An *Unary Implicational System* (UIS for short) Σ on S is a binary relation between $\mathcal{P}(S)$ and S : $\Sigma \subseteq \mathcal{P}(S) \times S$. An ordered pair $(A, b) \in \Sigma$ is called a *Σ-implication* whose *premise* is A and *conclusion* is b. It is written $A \rightarrow b$, meaning "A implies b". A subset $X \subseteq S$ *respects* a Σ-implication $A \rightarrow b$ when $A \subseteq X$ implies $b \in X$ (i.e. "if X contains A then X contains b"). $X \subseteq S$ is *Σ-closed* when X respects all Σ-implications, i.e $A \subseteq X$ implies $b \in X$ for every Σ-implication $A \rightarrow b$. The set of all Σ-closed sets forms a closure system $\mathbb{F}_\Sigma$ on S :

$$\mathbb{F}_\Sigma = \{X \subseteq S \,|\, X \text{ is } \Sigma\text{-closed}\} \tag{6}$$

Then we can associate to Σ a closure operator $\varphi_\Sigma = \varphi_{\mathbb{F}_\Sigma}$ which defines the closure of a subset $X \subseteq S$

$$\varphi_\Sigma(X) = \pi_\Sigma(X) \cup \pi^2_\Sigma(X) \cup \pi^3_\Sigma(X) \cup \ldots \text{ where} \tag{7}$$

$$\pi_\Sigma(X) = X \cup \bigcup \{b \,|\, A \subseteq X \text{ and } A \rightarrow b\} \tag{8}$$

An UIS Σ is a *generating system* of the closure system $\mathbb{F}_\Sigma$ (using Eq. (4)), and thus for the induced closure operator φ, and the induced lattice $(\mathbb{F}_\Sigma, \subseteq)$. When some UISs Σ and Σ' on S are generating systems for the same closure system, they are called *equivalent* (i.e. $\mathbb{F}_\Sigma = \mathbb{F}_{\Sigma'}$).

An UIS Σ is called *direct* or *iteration-free* if for every $X \subseteq S$, $\varphi(X) = \pi_\Sigma(X)$ (see Eq. (8)). An UIS Σ is *minimal* or *non-redundant* if $\Sigma \setminus \{X \rightarrow y\}$ is not equivalent to Σ, for all $X \rightarrow y$ in Σ. It is *minimum* if it is of least cardinality, i.e. if $|\Sigma| \leq |\Sigma'|$ for all UIS Σ' equivalent to Σ. A minimum UIS is trivially non-redundant, but the converse is not true in general. A minimal UIS is usually called a *basis* for the induced closure system (and thus for the induced lattice), and a *minimum basis* is then a basis of least cardinality.

The *canonical direct basis* described in [17] and denoted Σ_{cd}, has three main properties : directness, canonical and minimality properties.

In the literature, an implicational system (IS for short) Σ can also be defined as a binary relation on $\mathcal{P}(S)$. A Σ-implication is then an ordered pair $(A, B) \in \Sigma$, written $A \rightarrow B$, with $A, B \in \mathcal{P}(S)$. Generating systems (also called *covers*) and bases can be also defined for IS. In this case, there exists an unique minimum basis, called the *canonical basis*, also denoted the *Guigues and Duquenne basis* ([15]), enabling to get all the others minimum basis.

Other definitions and bibliographical remarks can be found in the survey of Caspard and Monjardet in [5].

Bordat's algorithm. One of the first algorithm generating the concept lattice is due to Chein [6] : concepts are generating from the inital context using submatrix computation algorithm. These first algorithms has then been improved using efficient methods for testing wether a concept has been already generated. The initial algorithms are Norris's algorithm [23], Next Closure of Ganter [11], Bordat's algorithm [3]. Norris's algorithm is an incremental algorithm, as algorithms in [13] and [27]. Next Closure algorithm defines the *lectic order* (extended the inclusion order) between concept for testing wether a concept has been already generated. Bordat's agorithm computes the Hasse diagram of the concept lattice, as algorithms in [4] and [27]. Nourine and Raynaud [24] use a sophisticated tree data structure to generated concepts with the best theoretical complexity.

```
Name : Concept-Lattice
Input: A context $K = (G, M, I)$
Output: The cover relation $(\mathbb{C}, \prec)$ of the concept lattice of $K$
begin
    $\mathbb{C} = \{(f(G), G)\}$;
    foreach $(A, B) \in \mathbb{C}$ not marked do
        F=cover $(K, (A, B))$;
        foreach $B' \in F$ do
            $A' = g(B')$;
            if $(A', B') \notin \mathbb{C}$ then add $(A', B')$ in $\mathbb{C}$;
            add the cover relation $(A, B) \prec (A', B')$
        end
        mark $(A, B)$
    end
    return $(\mathbb{C}, \prec)$
end
```

For our classification problem, Bordat's algorithm [3], or any algorithm generating the Hasse diagram, is the more appropriate since it generates the cover relation between concepts. Bordat's algorithm is issued from Theorem 1 that is redefined in Corollary 1.

Theorem 1 (Bordat [3]). *Let (A, B) and (A', B') two concepts of a context (G, M, I). Then $(A, B) \prec (A', B')$ if and only if B' is inclusion-maximal in the following set system F_A defined on G :*

$$F_A = \{f(a) \cap B \ : \ a \in M - A\} \tag{9}$$

Corollary 1 (Bordat [3]). *Let (A, B) be a concept. There is a one-to-one mapping between $Cover((A, B))$ and the inclusion-maximal subsets of F_A where :*

$$Cover((A, B)) = \{(A', B') \ : \ (A, B) \prec (A', B')\} \tag{10}$$

Bordat's algorithm in Algorithm `Concept-Lattice` computes all the concepts of $\mathbb{C}$ by computing $cover(A, B)$ for each concept (A, B), starting from the bottom concept $\bot = (f(G), G)$, until all concepts are generated. It is in $O(|\mathbb{C}||M|^{\alpha})$, with $2,5 \leq \alpha \leq 3$, since each concept is issued from the computation of $cover((A, B))$ that is in $O(|M|^{\alpha})$.

Algorithm `cover-objects` describe the two steps of the computation of $cover((A, B))$: the set system F_A has first to be generated in a linear time since F_A can be computed in an incremental way ; then inclusion-maximal subsets of F_A can easily be computed in $O(|M|^3)$, but the inclusion-maximal subsets problem is known to be resolved in $O(M|^{2,5})$ using sophisticated data structures ([10,22]).

```
Name : Cover-objects
Input: A context $K$ ; A concept $(A, B)$ of $K$
Output: The inclusion-maximal subsets of $F_A$
begin
    1. Compute $F_A$ : $F_A = \{f(a) \cap B \ : \ a \in M - A\}$;
    2. Compute $F$ : the maximal-inclusion subsets of $F_A$;
    return $F$
end
```

3 Extensions of Bordat's algorithm for recognition of noised symbols

Classification with a concept lattice. In our current works in [16], we use a concept lattice as classifier of noised symbol images. Each symbol X is giving by a vector of features $(x_i)_{i \leq n}$, denoted its *signature*, and extracted from the image of a symbol X using image analysis treatment. A class information $c(X)$ is also associated to each symbol.

The classification problem consists then in computing the class information of not classed and noised symbols. This problem is usually decomposed into two step : a *learning step* aiming at generating a classifier from a set of symbols as input ; a *classification step* aiming at classify noised symbols using the classifier. The noised symbol recognition problem takes as learning input a set of symbols given by their signature and their class ; and as classification symbols a set of noised symbols given by their signatures.

In the learning step, features of the signature are discretized into intervals in order to separate between symbols of different classes. The relation between the learning symbols (objects) and the features's intervals (attributes) forms a context $K = (G, M, I)$ and the classifier is then the cover relation of the concept lattice of K.

This concept lattice classifier is used to classify a noised symbol $X = (x_i)_{i \leq n}$ in a second step by navigation into the concept lattice. The navigation consists in a breath-first search in the concept lattice starting from the bottom concept, until a final concept is reached, using a choice criteria to choose according to the cover-relation between concepts. Class of the input noised symbol is then the class associated to the reached final concept, thus a concept is a final concept when it is composed of objects of the same class. A final concept is covered by the top concept. This classifier is based on three extensions of Bordat's algorithm :

1. We need attributes and not objects during the navigation in the concept lattice. Thus a first extension of Bordat's algorithm to compute the closure system on attributes instead if the concept lattice.
2. We explore by navigation only a small part of the concept lattice depending on the input symbols to classify. So, all concepts are not required. The second extension of Bordat's algorithm consists to on-line generate a concept only when it is reached by navigation.
3. Implication between attributes represents an efficient tools to describe attributes. The third extension of Bordat's algorithm is a generation of the canonical direct basis while generating concepts.

Generation of the closure system on attributes. In the context of recognition of noised symbols, the cover relation $(\mathbb{C}_M, \prec)$ of the closure system on attributes is sufficient for the navigation.

Let $A \in \mathbb{C}_M$ be a closed set. The cover of A in the lattice $(\mathbb{C}_M, \prec)$ is redefined as :

$$Cover(A) = \{A' \,:\, A' \prec A\} \tag{11}$$

Although the set system F_A is composed of subsets of objects of G, it is defined according to attributes of $M \setminus A$ (see 9). Let us consider the two following cases for a and a' two attributes of $M \setminus A$:

1. If $f(a) \cap B = f(a') \cap B$ then a and a' give raise to one subset $f(a) \cap B$ in F_A. Thus a and a' are equivalent in this case.
2. If $f(a) \cap B \neq f(a') \cap B$, then the two subsets $f(a) \cap B$ and $f(a') \cap B$ belong to F_A. The maximal-inclusion subsets of F_A are deduced from the inclusion relation between $f(a) \cap B$ and $f(a') \cap B$, and can be extended to a relation between a and a'.

To formalize the first case, we introduce an equivalence relation $\sim$ on attributes of $M \setminus A$. The second case can then be reformulated using an ordered relation $\lhd_A$ on the set of equivalence classes of $\sim$. This set is called the quotient set by $\sim$ and denoted $(M \setminus A)/_\sim$:

1. $\sim$ is an equivalence relation $\sim$ on $M \setminus A$ defined by

$$\forall a, a' \in M \setminus A,\ a \sim a' \iff f(a) \cap B = f(a') \cap B \tag{12}$$

2. $\lhd_A$ is the order relation defined on the quotient set $(M \setminus A)/\sim$ by :

$$\forall a, a' \in M \setminus A,\ \lfloor a \rfloor \lhd_A \lfloor a' \rfloor \iff f(a) \cap B \subseteq f(a') \cap B \tag{13}$$

Therefore, F_A can be be extended to an order relation on equivalence classes on attributes of $M \setminus A$. The following corollary extends Theorem 1 to an use of the only attributes, and gives raise to Algorithm `Closure-System` that generates the cover each closed set A using Algorithm `cover-attributes`.

Corollary 2. *Let $A \in \mathbb{C}_M$. There is a one-to-one mapping between $Cover(A)$ and the maximal elements of the poset $((M \setminus A)/_\sim, \triangleleft_A)$.*

```
Name : Closure-System
Input: A context $K = (G, M, I)$
Output: The cover relation $(\mathbb{C}_M, \prec)$ of the lattice $(\mathbb{C}_M, \subseteq)$
begin
    $\mathbb{C}_M = \{f(G)\}$;
    foreach $A \in \mathbb{C}_M$ not marked do
        F=cover-attributes $(K, A)$;
        foreach $X \in F$ do
            $A' = A + X$;
            if $A' \notin \mathbb{C}_M$ then add $A'$ in $\mathbb{C}_M$;
            add a cover relation $A \prec A'$
        end
        mark $A$
    end
    return $(\mathbb{C}_M, \prec)$
end
```

Algorithm `cover-attributes` is very similar to Algorithm `cover-attributes`, and can also be resolved in $O(|\mathbb{C}||M|^\alpha)$ using sophisticated data structure. However, the maximal elements of the inclusion relation $\triangleleft_A$ can be updated in a incremental way in $O(|M|^3)$ since equivalence classes for a closed set A are included in those of closed set successors of A. Therefore, Algorithm `cover-attributes` is in $O(|M|^3)$, and Algorithm `Closure-System` is in $O(|\mathbb{C}||M|^3)$, as Bordat's algorithm.

A similar poset introduced by Morvan and Nourine and denoted G'_A in [20] or $Pred^*_A$ in [21] is defined according to $B \setminus f(a)$ instead of $f(A) \cap B$. They state the following equivalent theorem :

Theorem 2 (Morvan, Nourine [20]). *Let $A \in \mathbb{C}_M$. There is a one-to-one mapping between $Cover(A)$ and $min(G'_A)$.*

They derive from this theorem a generation algorithm of "minimal interval extensions" based on a one-to-one mapping between theses extensions and the maximal chains of a closure system ordered by inclusion (i.e. a lattice). In another paper [21], this algorithm has been simplified to the distributive case (case where the concept lattice is distributive) in $O(|\mathbb{C}|)$: in this case, the $\triangleleft_A$'s similar poset can be computed in a post-treatment, and thus has not to be updated for each closed set. This generation is called the *strong simplicial elimination scheme* and corresponds to the decomposition process of a distributive lattice in intervals described by Markowski in [19] ; as to the duplication process of intervals introduced by Day in [7] and generalized to other duplications in [8,9,1].

```
Name : cover-attributes
Input: A context $K$ ; closed set $A \in \mathbb{C}_M$ of attributes
Output: The maximal elements of $((M \setminus A)/_\sim, \lhd_A)$.
begin
    1. Compute $(M \setminus A)/_\sim$ : the equivalences classes of $\sim$ on $M \setminus A$.
    2. Compute $\lhd_A$ : the inclusion relation on $(M \setminus A)/_\sim$;
    3. Compute $F$ : the maximal elements of $\lhd_A$
    return $F$
end
```

Extension to an on-line generation. Algorithm `On-Line-Closure-System` is an extension of Bordat's algorithm to on-line generate closed sets according to a choice criteria to select a closed set between *cover*(A). Since Bordat's algorithm generates concepts with a breath-first strategy, it has been adapted to a death-first strategy in a recursive way, and has initially to be called with $\varphi(\emptyset)$ as first closed set.

Let us notice that a death-first generation of all closed sets would consists in replacing `Select` X `in` F by the loop `ForEach` X `in` F to consider all subsets of F in the same way, thus uselessness of the else statement.

Algorithm `On-Line-Closure-System` is in $(O(|\mathbb{C}^{on-line}||M|^3)$ where $\mathbb{C}^{on-line}$ is the set of closed set on-line generated. This set depends of the symbols to classify : $|M| \leq |\mathbb{C}^{on-line}| \leq |\mathbb{C}|$

Table 3 illustrates the interest of an on-line generation for recognition of noised symbols.

	Learning	Recognition	Number of concepts
Total generation	430,2 sec	2 sec	3185
On-line generation	0,5 sec	9,8 sec	282

Tab. 1. Recognition of 10 noised symbols ; Learning with 25 not noised symbols

Extension to generation of the canonical direct basis. It is possible to extend relation $\lhd_A$ to be defined on $M \setminus A$ instead on the quotient set $(M \setminus A)_\sim$:

$$\forall a, a' \in M \setminus A,\ a \lhd_A a' \iff f(a) \cap B \subset f(a') \cap B \tag{14}$$

```
Name : On-Line-Closure-System
Input: A context $K = (G, M, I)$ ; a suborder $(\mathbb{C}'_M, \prec)$ of the cover relation $(\mathbb{C}, \prec)$ ;
       a closed set $A \in \mathbb{C}'_M$
begin
    if $A$ not marked then
        F=Cover-Attributes $(K, A)$;
        Select $X$ in $F$ according to the choice criteria;
        $A' = A + X$;
        add the cover relation $A \prec A'$;
        if $A' \notin \mathbb{C}'_M$ then add $A'$ in $\mathbb{C}'_M$;
        mark $A$;
    end
    else
        Let $F = \{A' \setminus A \,:\, A' \in \mathbb{C}'_M$ and $A \prec A'\}$;
        Select $X$ in $F$ according to the choice criteria;
    end
    if $A' \neq M$ then On-Line-Closure-System$(K, \mathbb{C}'_M, \prec, A')$;
end
```

It's important to notice that the relation $\lhd_A$ defined on $M \setminus A$ isn't an ordered relation since $a \lhd_A a'$ and $a' \lhd_A a$ for two $\sim$-equivalent attributes a and a'.

It is stated in [?] that $Pred^*_A$ (i.e. relation $\lhd_A$) is equivalent to the *dependance relation* δ defined for a lattice, and introduced in [?] (see also [?]). The *dependence relation* δ_X is defined on S, with $x, y \in S$ and $X \subset S$, by :

$$x \delta_X y \text{ if and only if } x \notin \varphi(X),\ y \notin \varphi(X) \text{ and } x \in \varphi(X + y) \qquad (15)$$

In [?], the authors state the equality between the canonical basis and five implicational systems issued from different works and satisfying various properties. One of the five implicational system is the dependence relation's basis on S is issued from the *dependence relation* :

$$\Sigma_{cd} = \{X + y \rightarrow x \,:\, x \delta_X y \text{ and } X \text{ is minimal for this property}\} \qquad (16)$$

Therefore, using relation $\lhd_A$, it is possible to compute the canonical direct basis Σ_{cd} of the closure set $\mathbb{C}'_M$ using Eq.16 as done by Algorithm `Canonical-Direct-Basis`. In particular $\lhd_\emptyset$ corresponds to the inclusion relation on M, and to unitary implicational rules in Σ_{cd} (i.e. rules with a singleton as premise).

The canonical direct basis Σ_{cd} can be computed in $(O(|\mathbb{C}_M||M|^3)$, thus a complexity of Algorithm `On-Line-Closure-System` in $(O(|\mathbb{C}_M||M|^3)$, as Bordat's algorithm.

4 Conclusion

We present in this paper three extensions of Bordat's algorithm. Although these extensions have been introduced for the noised symbol recognition problem, it can be implemented with any concept lattice.

```
Name : Canonical-Direct-Basis
Input: A context $K = (G, M, I)$
Output: The Cover relation of the lattice $(\mathbb{C}_M, \subseteq)$ ; The canonical direct basis $\Sigma_{cd}$
begin
    $\mathbb{C}_M = \{f(G)\}$;
    $\Sigma_{cd} = \emptyset$;
    Let $\lhd_\emptyset$ be the inclusion relation on $M$;
    foreach $(a, a') \in M^2$ such that $a \lhd_\emptyset a'$ do
        add $a' \to a$ in $\Sigma_{cd}$
    end
    foreach $A \in \mathbb{C}_M$ not marked do
        Let $F =$cover-attributes $(K, A)$;
        Let $\lhd_A =$ the inclusion relation on $M \setminus A$;
        foreach $X \in F$ do
            $A' = A + X$;
            foreach $(a, a') \in M \setminus A'$ such that $a \lhd_{A'} a'$ and $a \not\lhd_A a'$ do
                add $A + a' \to a$ in $\Sigma_{cd}$
            end
            if $A' \notin \mathbb{C}_M$ then add $A'$ in $\mathbb{C}_M$;
            add a cover relation $A \prec A'$
        end
        mark $A$
    end
    return $(\mathbb{C}_M, \prec)$ and $\Sigma_{cd}$
end
```

Acknowledgments to Philippe Sachot and Antoine Mercier for the implementation of the algorithms presented in this paper.

Références

1. K. Bertet and N. Caspard. Doubling convex sets in lattices : characterisations and recognition algorithms. *Order*, 19(2) :181–207, 2002.
2. G. Birkhoff and O. Frink. Representations of lattices by sets. *Transactions of the American Mathematical Society*, 64 :299–316, 1948.
3. J. Bordat. Calul pratique du treillis de galois d'une correspondance. *Math. Sci. Hum.*, 96 :31–47, 1986.
4. C. Carpineto and G. Romano. Galois : An order-theoretic approach to conceptual clustering. In *Proceedings of ICML'93*, pages 33–40, Amherst, July 1993.
5. N. Caspard and B. Monjardet. The lattice of closure systems, closure operators and implicational systems on a finite set : a survey. *Discrete Applied Mathematics*, 127(2) :241–269, 2003.
6. M. Chein. Algorithme de recherche des sous-matrices premières d'une matrice. *Bulletin Mathématique de la Sociologie Scientifique de la R.S. de Roumanie*, 1969.
7. A. Day. A simple solution of the word problem for lattices. *Canad. Math. Bull.*, 13 :253–254, 1970.

8. A. Day. Characterizations of finite lattices that are bounded-homomorphic images or sublattices of free lattices. *Canadian J. Math.*, 31 :69–78, 1979.
9. A. Day, J. Nation, and S. Tschanz. Doubling convex sets in lattices and a generalized semidistributivity condition. *Order*, 6 :175–180, 1989.
10. M. Fisher and A. Meyer. Boolean matrix multiplication and transitive closure. In 12^{th} *Annual Sympsosium on Switching and Automata Theory*, pages 129–131, 1971.
11. B. Ganter. Two basic algorithms in concept analysis. *Technische Hochschule Darmstadt (Preprint 831)*, 1984.
12. B. Ganter and R. Wille. *Formal concept analysis, Mathematical foundations.* Springer Verlag, Berlin, 1999.
13. R. Godin, R. Missaoui, and H. Alaoui. Learning algorithms using a Galois lattice structure. *Third International Conference on Tools for Artificial Intelligence, San Jose, Calif.*, pages 22–29, 1991.
14. J. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95 :5–18, 1986.
15. J. Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathematiques & Sciences Humaines*, 95 :5–18, 1986.
16. S. Guillas, K. Bertet, and J.-M. Ogier. Concept lattice classifier : a first step towards an iterative process of recognition of noised graphic objects. In *Fourth International Conference on Concept Lattices and their Applications (CLA'2006)*, pages 257–263, Tunisia, 2006.
17. B. M. K. Bertet. The multiple facets of the canonical direct basis, 2005.
18. D. Maier. *The Theory of Relational Databases.* Computer Sciences Press, 1983.
19. G. Markowski. Primes, irreducibles ans extremal lattices. *Order*, 9 :265–290, 1992.
20. M. Morvan and L. Nourine. Generating minimal interval extensions. Research Report 92-015, LIRMM, Montpellier, juin 1992.
21. M. Morvan and L. Nourine. Simplicial elimination shemes, extremal lattices and maximal antichains lattice. *Order*, 13(2) :159–173, 1996.
22. I. Munro. Efficient determination of the transitive closure of a directed graph. *Information Processing Letter*, pages 56–58, 1971.
23. E. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2), 1978.
24. L. Nourine and O. Raynaud. A fast algorithm for building lattices. In *Third International Conference on Orders, Algorithms and Applications*, Montpellier, France, august 1999.
25. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In L. S. Verlag, editor, *ICDT'99*, volume 1540, pages 398–416, 1999.
26. R. Taouil and Y. Bastide. Computing proper implications. *Proceedings of ICCS-2001 Internationnal Workshop on Concept Lattices-Based Theory, Methods and tools for Knowledge Discovery in Databases*, pages 290–303, 2001.
27. P. Valtchev, R. Missaoui, and P. Lebrun. A partition-based approach towards constructing galois (concept) lattices. *Discrete Mathematics*, 3(256) :801–829, 2002.

A new concise representation of frequent patterns through disjunctive search space

T. Hamrouni[1,2], I. Denden[1], S. Ben Yahia[1] and E. Mephu Nguifo[2]

[1] Faculty of Sciences of Tunis, Tunis, Tunisia.
{*tarek.hamrouni, sadok.benyahia*}*@fst.rnu.tn*
[2] CRIL-CNRS, IUT de Lens, Lens, France.
{*hamrouni, mephu*}*@cril.univ-artois.fr*

Abstract. The interest in a further pruning of the set of frequent patterns that can be drawn from real-life datasets is growing up. In fact, it is a quite survival reflex towards providing a manageably-sized and reliable knowledge. This fact is witnessed by the proliferation of what is called *concise representation* of frequent patterns. In this paper, we propose an exact concise representation that explores the *disjunctive search space* in addition to the conjunctive one, in contrast with almost all known concise representations which only focussed on the latter space. This representation required the definition of a new disjunctive closure operator. The latter operator partitions the search space into distinct disjunctive equivalence classes and, hence, makes possible to drastically reduce the number of handled patterns. Empirical evidences are presented about the relative size of the new representation *w.r.t.* those based on frequent closed, (closed) non-derivable and essential patterns, respectively.
Keywords: Frequent pattern, Concise representation, Disjunctive search space, Itemset.

1 Introduction and motivations

Within the traditional framework of association rule mining, managing the high number of frequent patterns extracted from real-life datasets becomes an important topic (1). A growing number of works hence explored the conjunctive search space to get out a nucleus of patterns, from which the remaining ones can be derived without information loss. Such an exploration was mainly motivated by the fact that the conjunctive operator – linking items – got the monopoly since the application of association rules in market basket analysis. Such a nucleus is better known as *exact concise representation*. Beyond expected high compactness rates, an exact concise representation should make possible to guess the frequency status of a pattern, and then to exactly retrieve its support when it is frequent enough. The main exact concise representations proposed are those based on frequent closed [1], non-derivable [2], closed non-derivable [3] (2) and essential patterns [4]. The first three representations also have the interesting property of being *true* (also called *perfect* in [4]) covers of frequent patterns, since their cardinality is always smaller than that of the frequent pattern set.

[1] Here we are mainly interested in itemsets as a pattern class.
[2] This representation simply gathers the set of closures of frequent non-derivable patterns. It is, hence, smaller in size terms than the previous two ones.

The main originality of the concise representation based on frequent essential patterns stands in the fact that it mainly explores the *disjunctive search space* where elements are characterized by their respective disjunctive supports, instead of conjunctive ones. It hence makes use of the inclusion-exclusion identities [5] to bridge both conjunctive and disjunctive search spaces. Nevertheless, in spite of such originality, this representation suffers from two major disadvantages:
1. It is not self-contained in the sense that the essential pattern set does not make possible by itself to decide whether a pattern is frequent or not. Hence, such a set has to be burdened by additional elements belonging to the positive border of the order ideal induced by the frequency constraint.
2. Several essential patterns can characterize the same set of objects and, therefore, they present a certain form of redundancy.
In this situation, finding a closure operator related to essential patterns would be of paramount importance to get a more reduced concise representation. Indeed, thanks to this operator, many essential patterns will be mapped into the same element within the disjunctive search space. Thus, the obtained representation will be more compact, especially for dense datasets. Furthermore, the simultaneous use of essential patterns and disjunctive closed ones can also ease the detection of their respective disjunctive equivalence classes and, hence, the traversal of the disjunctive search space. This can intensively be explored in many applications as done within the conjunctive search space thanks to their correspondences; minimal generators and closed patterns respectively (see [6] for a study). Indeed, these particular patterns are structurally localized within the associated lattice what gives them more semantics, contrary to other patterns numerically retained (like non-derivable patterns) independently from their localization.

The rest of the paper is arranged as follows. The next section recalls the key notions used throughout this paper. Section 3 describes the concise representation based on frequent essential patterns. The disjunctive closure operator as well as its main properties are detailed in Section 4, where a new disjunctive closure-based concise representation is also introduced. The empirical evidences about the utility of our approach are provided in Section 5. Section 6 discusses the main related work.

2 Key notions

In this section, we briefly sketch the key notions used in the remainder of this paper.

Definition 1. (EXTRACTION CONTEXT) *An extraction context is a triplet $\mathcal{K}$ = ($\mathcal{O}$, $\mathcal{I}$, $\mathcal{R}$), where $\mathcal{O}$ represents a finite set of objects, $\mathcal{I}$ is a finite set of items and $\mathcal{R}$ is a binary* (*incidence*) *relation* (*i.e.,* $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$). *Each couple* $(o, i) \in \mathcal{R}$ *expresses that the object* $o \in \mathcal{O}$ *contains the item* $i \in \mathcal{I}$.

Example 1. *In the remainder, we will consider the extraction context depicted by Table 1 with* $\mathcal{O} = \{1, 2, 3, 4, 5, 6, 7\}$ *and* $\mathcal{I} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$.

A pattern can be characterized by three kinds of supports as sketched by the following definition.

Definition 2. *[5]* (SUPPORTS OF A PATTERN) *Let* $\mathcal{K}$ = ($\mathcal{O}$, $\mathcal{I}$, $\mathcal{R}$) *be an extraction context. We distinguish three kinds of supports associated to a pattern* I:

	a	b	c	d
1	×			
2	×	×		
3	×		×	
4	×			×
5	×	×	×	
6	×	×		×
7	×		×	×

Table 1. An extraction context.

- Conjunctive support: *Supp*(I) = $\mid \{o \in \mathcal{O} \mid (\forall\, i \in I, (o, i) \in \mathcal{R})\} \mid$
- Disjunctive support: *Supp*$(\vee I)$ = $\mid \{o \in \mathcal{O} \mid (\exists\, i \in I, (o, i) \in \mathcal{R})\} \mid$
- Negative support: *Supp*$(\neg I)$ = $\mid \{o \in \mathcal{O} \mid (\forall\, i \in I, (o, i) \notin \mathcal{R})\} \mid$

A pattern I is said to be *frequent* if *Supp*(I) is greater than or equal to a user-specified minimum support threshold, denoted *minsup*. Since frequent patterns fulfill the order ideal property [7], the supersets of infrequent items will also be infrequent. The set of items $\mathcal{I}$ (and consequently the extraction context $\mathcal{K}$) will hence be considered as only containing frequent ones. Infrequent items will thus be pruned. Please also note that *Supp*$(\vee I) \geq$ *Supp*(I).

Given the respective disjunctive supports of a pattern's subsets, we are able to derive its conjunctive support using the *inclusion-exclusion identities* [5]. Furthermore, thanks to the *De Morgan's law*, we are even able to straightforwardly derive its negative support. Lemma 1 shows these important properties.

Lemma 1. (DERIVATION OF THE CONJUNCTIVE AND NEGATIVE SUPPORTS) *Let $I \subseteq \mathcal{I}$ be an arbitrary pattern. Its conjunctive and negative supports are respectively derived as follows:*

$$\begin{aligned} Supp(I) &= \sum_{\emptyset \subset I_1 \subseteq I} (-1)^{\mid I_1 \mid - 1}\, Supp(\vee I_1) \qquad (1) \\ Supp(\neg I) &= \mid \mathcal{O} \mid - Supp(\vee I) \qquad (2) \end{aligned}$$

Example 2. *Consider the extraction context of Table 1. Given the respective disjunctive supports of* bc*' subsets* [3]*, its conjunctive and negative supports are inferred as follows:*
• *Supp*(bc) = $(-1)^{|bc|-1}$ *Supp*($\vee$bc) + $(-1)^{|b|-1}$ *Supp*($\vee$b) + $(-1)^{|c|-1}$ *Supp*($\vee$c) = - *Supp*($\vee$bc) + *Supp*($\vee$b) + *Supp*($\vee$c) = - **5** + **3** + **3** = **1**.
• *Supp*($\neg$bc) = $|\mathcal{O}|$ - *Supp*($\vee$bc) = **7** - *Supp*($\vee$bc) = **7** - **5** = **2**.

3 Frequent essential pattern-based concise representation

The next definition presents the frequent essential patterns. These patterns constitute the core of the concise representation which motivates ours (*cf.* Section 1).

Definition 3. *[4]* (FREQUENT ESSENTIAL PATTERN) *Let $\mathcal{K}$ = $(\mathcal{O}, \mathcal{I}, \mathcal{R})$ be an extraction context and $I \subseteq \mathcal{I}$. I is an essential pattern* iff *Supp*$(\vee I) \neq \max\{$*Supp*$(\vee I \backslash i) \mid i \in I\}$. *An essential pattern I is also frequent if Supp$(I) \geq$ minsup.*

Example 3. *Consider the extraction context of Table 1 for minsup* = 1. ad *is not an essential pattern since Supp*($\vee$ad) = *Supp*($\vee$a) = **7**. *Whereas* bc *is an essential pattern since Supp*($\vee$bc) = **5** $\neq \max\{$*Supp*($\vee$b), *Supp*($\vee$c)$\}$ *since Supp*($\vee$b) = *Supp*($\vee$c) = **3**. bc *is also frequent since Supp*(bc) = **1** $\geq$ *minsup.*

[3] We use a separator-free form for the sets, *e.g.*, the set bc stands for {b, c}.

The set of frequent essential patterns, denoted $\mathcal{FEP}_{\mathcal{K}}$, was proven in [4] to be an order ideal in $(\mathbf{2}^{\mathcal{I}}, \subseteq)$. The following theorem presents the frequent essential pattern-based concise representation. $\mathcal{BD}^{+}(\mathcal{FP}_{\mathcal{K}})$ denotes the set of maximal frequent patterns, which is used to detect the frequency status of an arbitrary pattern.

Theorem 1. *[4] The set $\mathcal{FEP}_{\mathcal{K}}$ of frequent essential patterns increased by $\mathcal{BD}^{+}(\mathcal{FP}_{\mathcal{K}})$ constitutes an exact concise representation of the set of frequent patterns.*

It is worth noting that in [8], this representation was shown not to be perfect, contrary to the authors' claim.

4 New disjunctive closure-based concise representation

Here we detail the main constructs related to the disjunctive closure operator [8], which will make possible to map several essential patterns into a unique element within the disjunctive search space. This is the starting point of our new concise representation.

4.1 The disjunctive closure operator

Let us start by defining the disjunctive closure operator.

Definition 4. (DISJUNCTIVE CLOSURE OPERATOR) *Let $\mathcal{K}$ = $(\mathcal{O}, \mathcal{I}, \mathcal{R})$ be an extraction context. The disjunctive closure operator h is defined as follows:*
$h : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{I})$
$I \mapsto h(I) = \{i \in \mathcal{I} \mid (\forall\, o \in \mathcal{O})\ ((o, i) \in \mathcal{R}) \Rightarrow (\exists\, i_1 \in I)((o, i_1) \in \mathcal{R})\}$.

Roughly speaking, the disjunctive closure $h(I)$ of a pattern I is equal to the maximal set of items which *only* appear in the objects that contain at least an item of I.

Example 4. *Given the extraction context depicted by Table 1, the pattern* bc *is a disjunctive closed pattern since it is equal to the maximal set of items only contained in the set of objects where* b or c *appears, i.e.,* $\{\mathbf{2}, \mathbf{3}, \mathbf{5}, \mathbf{6}, \mathbf{7}\}$. *Hence,* $h($bc$)$ = bc. *While* acd *is not a disjunctive closed pattern since* b *only appears in the set of objects where at least an item of* acd *appears. In fact,* $h($acd$)$ = abcd.

Actually, Definition 4 gives an explicit expression of the disjunctive closure operator, free from the connection operators linking $\mathcal{P}(\mathcal{I})$ and $\mathcal{P}(\mathcal{O})$. This definition structurally characterizes the disjunctive closure of any pattern X and, hence, allows to straightforwardly compute it from any extraction context. To the best of our knowledge, our work is the first one allowing the extraction of a concise representation of frequent patterns based on a disjunctive closure operator, and, hence exploring the disjunctive search space. We will denote by $\mathcal{DCP}_{\mathcal{K}}$ the set of disjunctive closed patterns extracted from a context $\mathcal{K}$. Thanks to the closure operator h, the disjunctive search space is partitioned into distinct disjunctive equivalence classes. In the latter classes, disjunctive closed (*resp.* essential) patterns are the largest (*resp.* minimal) elements, *w.r.t.* set inclusion.

The following propositions allow to establish the relation between the smallest disjunctive closed pattern containing a pattern I and $h(I)$.

Proposition 1. *Let $I \subseteq \mathcal{I}$. $h(I)$ is the smallest disjunctive closed pattern containing I:*

$$h(I) = min_{\subseteq}\{f \in \mathcal{DCP}_{\mathcal{K}} \mid I \subseteq f\}.$$

Proposition 2. *Let $I \subseteq \mathcal{I}$. Supp($\vee$ I) = Supp($\vee$ $h(I)$).*

Proposition 3 makes possible to deduce the disjunctive closure of a pattern using the disjunctive closure of one of its subsets, while Proposition 4 establishes the link between disjunctive closed patterns and frequent essential patterns.

Proposition 3. *Let $X \subseteq \mathcal{I}$ and $Y \subseteq \mathcal{I}$ be two patterns. We then have:*

$$(X \subseteq Y \subseteq h(X)) \Rightarrow (h(Y) = h(X)).$$

Proposition 4. *Let $I \subseteq \mathcal{I}$ and $\mathcal{FP}_{\mathcal{K}}$ be the set of frequent patterns. We then have:*

$$(I \in \mathcal{FP}_{\mathcal{K}}) \Rightarrow (\exists\ f \in \mathcal{DCP}_{\mathcal{K}} \text{ and } I_1 \in \mathcal{FEP}_{\mathcal{K}} \text{ s.t. } h(I_1) = h(I) = f \text{ and } I_1 \subseteq I).$$

Proof. *(Sketch) The proof is based on the fact that the set $\mathcal{FEP}_{\mathcal{K}}$ is an order ideal in $(\mathbf{2}^{\mathcal{I}}, \subseteq)$ whose elements are the minimal ones in their associated disjunctive equivalence classes.*

In the remainder of the paper, we will denote by $\mathcal{EDCP}_{\mathcal{K}}$ ($\mathcal{E}$ssential $\mathcal{D}$isjunctive $\mathcal{C}$losed $\mathcal{P}$atterns) the subset of $\mathcal{DCP}_{\mathcal{K}}$ whose elements have at least a frequent essential pattern as generator. Thanks to Proposition 4, it is easy to show that the disjunctive closures of the patterns belonging to $\mathcal{BD}^+(\mathcal{FP}_{\mathcal{K}})$ are contained in $\mathcal{EDCP}_{\mathcal{K}}$.

Example 5. *Consider the context of Table 1. Within the disjunctive lattice sketched by Figure 1, different sets of patterns are indicated. The essential patterns are shown with bold letters, while the disjunctive closed patterns are underlined. The set $\mathcal{FEP}_{\mathcal{K}}$ induces an order ideal structure, as shown in Figure 1 for minsup = $\mathbf{1}$. Let $\mathcal{BD}^-(\mathcal{FEP}_{\mathcal{K}})$ be the negative border of $\mathcal{FEP}_{\mathcal{K}}$ equal to $\min_{\subseteq}\{I \in \mathcal{P}(\mathcal{I}) \setminus \mathcal{FEP}_{\mathcal{K}}\}$. The elements belonging to this border are in italic. An example of a disjunctive equivalence class, induced by the disjunctive closure operator, is also sketched. Its minimal element is the essential pattern* `a` *and its largest one is the disjunctive closed pattern* `abcd`*. Please note that if, for example, a pattern is in bold letters and is also underlined, then this means that it is both an essential pattern and a disjunctive closed one. As an indication, the patterns belonging to $\mathcal{BD}^+(\mathcal{FP}_{\mathcal{K}})$ are encircled.*

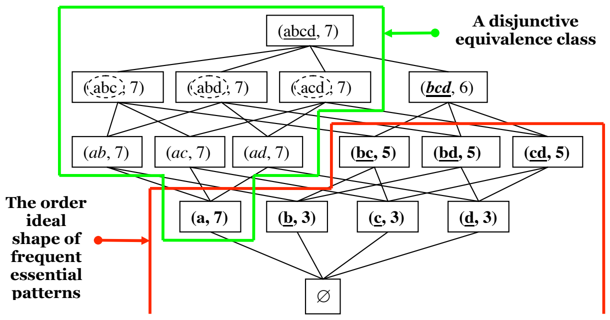


Fig. 1. The associated disjunctive lattice where each node contains a disjunctive pattern with its disjunctive support.

4.2 New disjunctive closure-based concise representation

It is commonly known that the definition of a concise representation is closely related to the way the whole set of frequent patterns will be generated starting from its elements. Suppose we have at hand the set $\mathcal{EDCP}_{\mathcal{K}}$ where each element is provided with its disjunctive support (as it is the case in [8]). We need to analyze the "tools" that will be of help in such a regeneration process. To the best of our knowledge, only the formula shown in Lemma 1 makes the link between the disjunctive support of a pattern and its conjunctive one. This formula requires knowing beforehand the disjunctive supports of the subsets of a given candidate to be able to compute its conjunctive support. Hence, an APRIORI-like regeneration is naturally advocated. This manner of regeneration consists in finding the conjunctive supports of $\mathbf{1}$-patterns, $\mathbf{2}$-patterns, and so on.

Let X be a pattern to which we are interested in retrieving its conjunctive support. Reaching X is conditioned by the fact that all its subsets (and more precisely, the immediate ones) are proven to be frequent. Indeed, the set of frequent patterns is an order ideal [7]. Hence, if a subset of X is infrequent, then X will necessarily be infrequent. Assume now that all subsets of X are frequent. At this step, the main information we have about each subset consists in its disjunctive closure (*cf.* Proposition 1) and, consequently, its different supports (*cf.* Lemma 1). If X is included in the closure of one of its immediate subsets, then we have its disjunctive closure and, hence, its disjunctive support (*cf.* Proposition 3). We can thus compute its conjunctive support. Please note that in this case, X is obviously not an essential pattern. If X is included in none of its subsets' closures, then X is necessarily an essential pattern. However, the closure of X is required to correctly compute its conjunctive support and then deduce if X is frequent or not. Nevertheless, how can we ensure that such a closure belongs to $\mathcal{EDCP}_{\mathcal{K}}$? Indeed, X can be an *infrequent* pattern and, at the same time, the *unique* generator of its disjunctive equivalence class. Hence, its closure will necessarily not be in $\mathcal{EDCP}_{\mathcal{K}}$ [4]. This important part was missed in [8], what motivates a careful scrutiny to correct the representation and make it really exact.

At this step of the treatment, to correctly regenerate the whole set of frequent patterns, it is clear that we need the disjunctive closures of frequent patterns (*i.e.*, $\mathcal{EDCP}_{\mathcal{K}}$), augmented by the closures *uniquely* generated by essential patterns belonging to the negative border of $\mathcal{FEP}_{\mathcal{K}}$. These latter closures do not belong to $\mathcal{EDCP}_{\mathcal{K}}$, but they bring key information when an infrequent essential pattern is reached. They are also necessarily sufficient because once an infrequent pattern is discovered all its supersets will not be treated. Hence, an important result is that $\mathcal{EDCP}_{\mathcal{K}}$ is not sufficient to ensure the exact regeneration of the whole frequent pattern set, what makes the claim of the authors in [8] incorrect. As characterized in the remainder, some closures should then be added to ensure that some candidates will not be erroneously considered as frequent whereas they are actually infrequent. These closures will form the set $\mathcal{ADCP}_{\mathcal{K}}$ ($\mathcal{A}$dded $\mathcal{D}$isjunctive $\mathcal{C}$losed $\mathcal{P}$atterns). An interesting question will be: how can we reduce the cardinality of $\mathcal{ADCP}_{\mathcal{K}}$ without affecting the exact regeneration of the whole frequent pattern set?

[4] If X is not the unique essential pattern of its disjunctive equivalence class $\mathcal{C}$, then its closure can belong to $\mathcal{EDCP}_{\mathcal{K}}$ if $\mathcal{C}$ contains at least a frequent pattern.

Let X be an infrequent essential pattern belonging to $\mathcal{BD}^-(\mathcal{FEP}_\mathcal{K})$. Let us have a look at the formula establishing the link between the conjunctive and disjunctive supports:

$$Supp(X) = \sum_{\emptyset \subset X' \subseteq X} (-1)^{|X'|-1} Supp(\vee X') = (-1)^{|X|-1} Supp(\vee X) + \sum_{\emptyset \subset X' \subset X} (-1)^{|X'|-1} Supp(\vee X').$$

Suppose that $|X|$ is even. Hence, $(-1)^{|X|-1} = \mathbf{-1}$. Assume now that we did not compute the disjunctive closure f of X. Then, two cases can arise: either X is covered by at least an element in $\mathcal{EDCP}_\mathcal{K}$ or is not covered at all (*i.e.*, $\forall f' \in \mathcal{EDCP}_\mathcal{K}, X \nsubseteq f'$). In the latter case, it is obvious that X is infrequent (*cf.* Proposition 4). Let us analyze the former case. Let f_1 be the smallest closure in $\mathcal{EDCP}_\mathcal{K}$ covering X. It is clear that $f \subseteq f_1$ (otherwise, the closure of X will never be f) [5]. Hence, $Supp(\vee f_1) \geq Supp(\vee f) = Supp(\vee X)$. Hence, if we use $Supp(\vee f_1)$ in the formula instead of $Supp(\vee X)$, the support value we obtain will be lower than or equal to the exact support of X [6]. This does not affect the final decision about the frequency status of X since it is infrequent and the possible decrease of its support will maintain its infrequency status. Hence, if X is an infrequent pattern of even size belonging to $\mathcal{BD}^-(\mathcal{FEP}_\mathcal{K})$, we need not compute its disjunctive closure, what consists in a very interesting pruning.

Example 6. *Consider the extraction context depicted by Table 1 for minsup =* $\mathbf{2}$. *Applying an extraction process, we obtain* $\mathcal{EDCP}_\mathcal{K} = \{(b, 3), (c, 3), (d, 3), (abcd, 7)\}$, *where each couple represents a disjunctive closed pattern and its disjunctive support. Let us regenerate the set of frequent patterns. We begin by* $\mathbf{1}$*-patterns, i.e.,* a*,* b*,* c *and* d*. The smallest closure containing* a *is* abcd*. Hence, its disjunctive support is equal to* $\mathbf{7}$*, which also corresponds to its conjunctive support. It is the same for the remaining* $\mathbf{1}$*-patterns. Thus, we find that their associated conjunctive supports are respectively equal to* $\mathbf{7}$*,* $\mathbf{3}$*,* $\mathbf{3}$ *and* $\mathbf{3}$*. We hence have the four candidates frequent. We then handle candidate* $\mathbf{2}$*-patterns. Consider the case of* bc *whose subsets are proven to be frequent. The smallest closure in* $\mathcal{EDCP}_\mathcal{K}$ *containing* bc *is* abcd*. However,* abcd *is not the actual closure of* bc*. Nevertheless, this does not affect the final decision about the frequency status of* bc*. Indeed, three cases should be distinguished: (i) if* bc *was frequent, hence its closure must belong to* $\mathcal{EDCP}_\mathcal{K}$*, (ii) if* bc *is not covered by the elements of* $\mathcal{EDCP}_\mathcal{K}$ *then* bc *is necessarily infrequent, otherwise, (iii) since* $|bc| = \mathbf{2}$*, then* $(-1)^{|bc|-1} = \mathbf{-1}$ *and hence taking a largest closure (i.e.,* abcd*), instead of the actual one (i.e.,* bc*) will decrease the result obtained thanks to Formula (1) (cf. Lemma 1), and, hence,* bc *will always be considered as infrequent and no status change can arise. Thus, the closure of* bc *is not required in the representation when* bc *is infrequent. Note that the application of Formula (1) is required independently from the frequency status of* bc *since we cannot guess its status beforehand only if it contains an infrequent subset what is not the case here.*

Unfortunately, such a pruning cannot be applied when X is of odd size. Indeed, in this case, $(-1)^{|X|-1} = \mathbf{+1}$. Thus, using $Supp(\vee f_1)$ instead of $Supp(\vee X)$ will

[5] f can be equal to f_1 if it also has a frequent essential pattern as generator.

[6] The computation of the conjunctive support of X is inevitable since we cannot beforehand predict whether it is frequent or not.

probably lead to the increase of *Supp*(X). Consequently, if X is infrequent and we augment its conjunctive support, then this may lead to a support value greater than or equal to *minsup* what clearly falsifies its frequency status. In this situation, we can further reduce the cardinality of $\mathcal{ADCP}_{\mathcal{K}}$ by only maintaining the closure f of X if it is included in at least an element of $\mathcal{EDCP}_{\mathcal{K}}$. Indeed, a pattern X is eligible to be frequent only if it is covered by a pattern of $\mathcal{EDCP}_{\mathcal{K}}$ (*cf.* Proposition 4). This can simply be done once f is computed by set inclusion operations with maximal elements of $\mathcal{EDCP}_{\mathcal{K}}$.

Example 7. *Now consider the context of Table 1 for minsup =* $\mathbf{1}$. $\mathcal{EDCP}_{\mathcal{K}} = \{(\mathsf{b}, 3), (\mathsf{c}, 3), (\mathsf{d}, 3), (\mathsf{bc}, 5), (\mathsf{bd}, 5), (\mathsf{cd}, 5), (\mathsf{abcd}, 7)\}$. *As in the previous example, we begin by* $\mathbf{1}$*-patterns, i.e.,* a, b, c *and* d. *We find that their associated conjunctive supports are respectively equal to* $\mathbf{7}$, $\mathbf{3}$, $\mathbf{3}$ *and* $\mathbf{3}$. *We then treat candidate* $\mathbf{2}$*-patterns and we find that the different candidates are frequent. We now reach candidate* $\mathbf{3}$*-patterns. The unique candidate is* bcd *since all its subsets are proven to be frequent.* bcd *hence fulfills the order ideal property of frequent patterns. It is also not contained in the closure of its subsets (cf. Figure 1).* bcd *is hence an essential pattern. If we will apply the same regeneration process to* bcd, abcd *will be considered as the disjunctive closure of* bcd *since it is the smallest one in* $\mathcal{EDCP}_{\mathcal{K}}$ *containing it. The conjunctive support of* bcd *will then be equal to* $\mathbf{1}$. *However, this is not true because* abcd *is not the actual disjunctive closure of* bcd. *The latter should be equal to* bcd. *Since* $|\mathsf{bcd}| = \mathbf{3}$, $(\text{-}1)^{|\mathsf{bcd}|-1} = \mathbf{+1}$ *and hence taking a largest closure (i.e.,* abcd*), instead of the actual one (i.e.,* bcd*), will augment the conjunctive support of* bcd, *actually equal to* $\mathbf{0}$, *which shifts its status from infrequent to frequent. The problem arises because* $\mathcal{EDCP}_{\mathcal{K}}$ *only contains closures having at least a frequent essential pattern as generator. This is not the case of* $h(\mathsf{bcd})$ *equal to* bcd *whose unique generator is obviously* bcd. *Such a closure necessarily does not belong to* $\mathcal{EDCP}_{\mathcal{K}}$ *since* bcd *is infrequent (its conjunctive support is equal to* $\mathbf{0}$*). Hence, its closure must be added to the representation to ensure not including* bcd *with the set of frequent patterns during the regeneration process.*

We now give the formal definition of the set $\mathcal{ADCP}_{\mathcal{K}}$ that ensures the new representation being exact.

Definition 5. *Let* $\mathcal{EP}_{\mathcal{K}}$ *be the set of the essential patterns that can be extracted from a context* $\mathcal{K}$. *The set* $\mathcal{ADCP}_{\mathcal{K}}$ *is defined as follows:* $\mathcal{ADCP}_{\mathcal{K}} = \{h(X) \mid (X \in \mathcal{BD}^{-}(\mathcal{FEP}_{\mathcal{K}}) \bigcap \mathcal{EP}_{\mathcal{K}}) \wedge ((-1)^{|X|} = \text{-}1) \wedge (\forall X' \subseteq \mathcal{I}, h(X') = h(X) \Rightarrow \textit{Supp}(X') < \textit{minsup}) \wedge (\exists f \in \mathcal{EDCP}_{\mathcal{K}} \textit{ s.t. } h(X) \subset f)\}$.

To summarize, $\mathcal{ADCP}_{\mathcal{K}}$ contains closures generated by infrequent essential patterns of odd sizes belonging to $\mathcal{BD}^{-}(\mathcal{FEP}_{\mathcal{K}})$. These closures have all their corresponding essential patterns infrequent and are covered by at least one element of $\mathcal{EDCP}_{\mathcal{K}}$. It is important to mention that in $\mathcal{ADCP}_{\mathcal{K}}$, we did not consider the disjunctive closures of *infrequent non*-essential patterns belonging to $\mathcal{BD}^{-}(\mathcal{FEP}_{\mathcal{K}})$ since they are already included in $\mathcal{EDCP}_{\mathcal{K}}$ (*cf.* Proposition 3).

The concise representation $\mathcal{EDCP}_{\mathcal{K}} \bigcup \mathcal{ADCP}_{\mathcal{K}}$ will be denoted $\mathcal{DCP}_{\mathcal{K}}\textit{\_rep}$.

Theorem 2. $\mathcal{DCP}_{\mathcal{K}}\textit{\_rep}$ *is an exact concise representation of* $\mathcal{FP}_{\mathcal{K}}$.

The proof of Theorem 2 can be treated as a naive algorithm for deriving frequent patterns and their associated supports.

In addition to the exact retrieval of frequent patterns as well as their various supports, $\mathcal{DCP}_{\mathcal{K}}\_rep$ presents three other main properties:
1. Homogeneity: $\mathcal{DCP}_{\mathcal{K}}\_rep$ only involves disjunctive closed patterns (*vs.* $\mathcal{FEP}_{\mathcal{K}} \bigcup \mathcal{BD}^{+}(\mathcal{FP}_{\mathcal{K}})$). Hence, it ensures the homogeneity of the representation since all its elements are provided with the same kind of support; the disjunctive one. They also have the same structural properties. Indeed, they are the top elements of their associated equivalence classes within the disjunctive search space.
2. Small size: In [8], the size of $\mathcal{EDCP}_{\mathcal{K}}$ is shown to be significantly smaller than those of the best known concise representations. In addition, the size of $\mathcal{ADCP}_{\mathcal{K}}$ is very small since its elements must fulfill many easy-to-check constraints. Hence, the size of $\mathcal{DCP}_{\mathcal{K}}\_rep$ will be, in most cases, smaller than those of the other representations.
3. Low regeneration cost: It is worth mentioning that our concise representation allows retrieving the conjunctive support faster than from (closed) non-derivable patterns [2, 3]. Indeed, for a pattern X *s.t.* $|X| = n$, the retrieval process of *Supp*(X) from these representations requires the costly evaluation of $\mathbf{2}^n$ deduction rules based on Bonferroni-inequalities [9]. The computation cost for inferring supports is then awfully high. While the retrieval of *Supp*(X) from our concise representation only needs to evaluate a unique inclusion-exclusion identity. Furthermore, it allows the straightforward retrieval of the disjunctive and negative supports of frequent patterns.

5 Experimental results

We compare, through various experiments, the size of our concise representation to those of the exact ones based on frequent closed, (closed) non-derivable and essential patterns. This is done in the most critical cases, *i.e.*, for strongly correlated datasets [7]. Indeed, within such datasets, the ratio between the cardinality of the frequent pattern set and those of concise representations is high. Thus, we are in the most interesting cases. Moreover, equivalence classes extracted from sparse datasets are often reduced to the associated generators and cannot be compacted anymore. This makes the size reduction rates brought by concise representations meaningless in such datasets. Due to lack of space, we only summarize the main results in this section.

All experiments were carried out on a PC equipped with a 1.73GHz Centrino Duo Core and 2GB of main memory, and running the Linux version Fedora Core 6 (with 2GB of swap memory). Results are shown in Table 5. The abbreviation "$\mathcal{FP}_{\mathcal{K}}\_set$"(*resp.* "$\mathcal{FCP}_{\mathcal{K}}\_rep$" [8], "$\mathcal{NDP}_{\mathcal{K}}\_rep$", "$\mathcal{CNDP}_{\mathcal{K}}\_rep$", and "$\mathcal{FEP}_{\mathcal{K}}\_rep$") is used to stand for the set of frequent patterns (*resp.* frequent closed, non-derivable, closed non-derivable and essential pattern-based concise representation). It is important to note that in the experimental results given in [3], the authors have chosen a specific interval of *minsup* for each dataset to extract $\mathcal{CNDP}_{\mathcal{K}}\_rep$. However, we noticed that their program abruptly comes to an end with an execution error beyond these intervals. Therefore, we use the symbol "-" to designate a case where an execution error occurred. At a glance, we can also deduce the following assertions:
1. Necessity to set up concise representations: Indeed, their respective sizes *w.r.t.* that

[7] These datasets are available at: *http://fimi.cs.helsinki.fi/data.*
[8] Source codes for extracting frequent (closed) patterns are available at: *http://fimi.cs.helsinki.fi/src.*

of the set of frequent patterns clearly show their utility and potential benefits. In particular, even for high *minsup* values, the cardinality of the introduced concise representation is considerably reduced.

2. Effectiveness of the proposed concise representation: Indeed, for CHESS, CONNECT and PUMSB datasets, the size of $\mathcal{DCP}_{\mathcal{K}}\_rep$ is significantly reduced compared to those of the remaining concise representations, while offering different kinds of patterns' supports.

3. Scalability of $\mathcal{DCP}_{\mathcal{K}}\_rep$: It is easily observable that, in most cases, the cardinality of $\mathcal{DCP}_{\mathcal{K}}\_rep$ is less sensible to the variation of *minsup* than those of the other concise representations.

4. Absence of an outstanding concise representation: For example, in some cases, the size of $\mathcal{DCP}_{\mathcal{K}}\_rep$ is slightly greater than the size of the other concise representations (*e.g.*, MUSHROOM for *minsup* = $\mathbf{5}\%$).

***minsup* (%)**	$\lvert\mathcal{FP}_{\mathcal{K}}\_set\rvert$	$\lvert\mathcal{FCP}_{\mathcal{K}}\_rep\rvert$	$\lvert\mathcal{NDP}_{\mathcal{K}}\_rep\rvert$	$\lvert\mathcal{CNDP}_{\mathcal{K}}\_rep\rvert$	$\lvert\mathcal{FEP}_{\mathcal{K}}\_rep\rvert$	$\lvert\mathcal{DCP}_{\mathcal{K}}\_\mathbf{rep}\rvert$
			CONNECT			
90	27, 127	3, 486	199	177	398	**22**
70	4, 129, 839	35, 875	545	491	1, 710	**161**
50	88, 324, 400	130, 112	1, 397	-	5, 063	**589**
30	1, 331, 673, 367	460, 356	3, 221	-	14, 083	**1, 986**
			MUSHROOM			
40	565	140	146	117	151	**91**
20	53, 583	1, 197	1, 143	731	1, 258	**941**
10	574, 431	4, 885	4, 347	2, 655	6, 530	**5, 457**
5	3, 755, 511	12, 843	11, 569	6, 546	24, 407	**20, 554**
			CHESS			
90	622	498	95	93	118	**43**
70	48, 731	23, 892	684	669	1, 482	**420**
50	1, 272, 932	369, 450	3, 425	3, 341	14, 272	**1, 971**
30	37, 282, 962	5, 316, 467	15, 147	-	147, 777	**8, 824**
			PUMSB			
90	2, 607	1, 467	586	460	788	**318**
80	142, 156	33, 308	3, 642	2, 136	6, 251	**1, 079**
70	2, 698, 654	241, 259	7, 875	4, 564	18, 318	**2, 143**
60	19, 529, 991	1, 074, 627	21, 323	-	54, 644	**5, 550**
50	165, 903, 540	7, 121, 264	47, 764	-	232, 581	**11, 551**

Table 2. Size of the different concise representations for benchmark datasets.

6 Discussion

First of all, let us make an alignment between the disjunctive search space and the conjunctive one. We will hence find that an essential pattern is the mapping of the concept of *minimal generator* (*aka key pattern* and *free-set* in the literature, see [6] for references) when the conjunctive search space is considered. While the disjunctive closed patterns are the mapping of conjunctive ones [1].

The concepts of essential and disjunctive closed patterns can be considered as particular cases of *composite items* [10] where the disjunction of items is used to compose new items, the composite ones. This is an attempt towards making useful infrequent items in some applications. For example, consider the context of Table 1 and let *minsup* = $\mathbf{4}$, b and c are hence infrequent items since their support is equal to $\mathbf{3}$. Nevertheless, the support of $b \vee c$ is equal to $\mathbf{5}$ and, hence, *Supp*($b \vee c$) $\geq$ *minsup*. $b \vee c$ will be considered as a new item (a composite one) even if, actually it is composed of two items. It will be used during the mining process since it is frequent what makes b and c useful.

It is important to make the link between our work and that of Zhao *et al*. Indeed, in [11], the authors proposed connection operators to link $\mathcal{P}(\mathcal{I})$ and $\mathcal{P}(\mathcal{O})$ for the case of disjunctive Boolean expressions. Nevertheless, their definition of the operator linking $\mathcal{P}(\mathcal{O})$ to $\mathcal{P}(\mathcal{I})$ depends on that ensuring the opposite direction and was not independently given from any other operator. Furthermore, they neither proposed the expression of the resulting closure operator nor carried out a thorough analysis of inherent structural properties.

The disjunction operator (*i.e.*, the operator $\vee$) has also been used to define some concise representations only exploring the conjunctive search space, like those based on disjunction-free sets and (generalized) disjunction-free generators [12] (9). This required the introduction of what is called *disjunctive rule*. Such a rule has a premise part composed by a conjunction of items and a conclusion part, distinct from the premise one, containing a specified number of items linked using the disjunction operator [12].

Some works [13, 14] were interested in using disjunction within association rules to define what is called generalized association rules. These rules grasped the interest of many researchers since they offer wealthier types of knowledge in many applications. In addition to the inclusive disjunction operator, *i.e.*, the operator $\vee$, the authors in [13] were also interested in the exclusive disjunction operator, denoted $\oplus$. In [14], the author mainly focusses on association rules having conclusions containing mutually exclusive items, *i.e.*, the presence of one of them leads to the absence of the others, what is expressed in [13] using the operator $\oplus$. Other forms of generalized association rules were also described in [15].

7 Conclusion and future work

In this paper, we presented a new disjunctive closure operator as well as its main properties. Based on this operator, we introduced a new concise representation which corrects the claim of [8] where the associated representation can miss some cases. This required the addition of few further elements what ensures the correctness of the whole regeneration process of frequent patterns. In addition to interesting compactness rates, our concise representation allows a straightforward computation of the disjunctive and negative supports. The experimental results showed that, in most cases, its size is significantly smaller than those of the best known concise representations. It is worth noting that our approach can easily be extended when negative items are handled.

Other avenues for future work mainly address the following points: First, due to space limitations here, we intend to address as next step the complexity time issue (generation and derivability) of our representation *vs.* those of the literature. In this respect, other algorithms for mining conjunctive closed patterns could be adapted to disjunctive ones, both breadth-first search algorithms and depth-first ones. Second, a structural characterization of disjunctive closed patterns *w.r.t.* existing frameworks like the k-free sets [12] will be done. Another important task consists in overcoming the lack in the literature of semantics' studies related to concise representations. The study of the possible extension of our representation to other pattern classes should also be examined. Finally, the extraction of generalized association rules will be thoroughly

[9] We did not use these representations in our experiments since $\mathcal{NDP}_{\mathcal{K}}\_rep$ (and consequently, $\mathcal{CNDP}_{\mathcal{K}}\_rep$) is shown in [2] to provide better results.

addressed. Indeed, setting up a theoretical framework that includes different kinds of operators is of paramount importance for jumping beyond standard association rules.

Acknowledgements: We thank the anonymous reviewers for their helpful comments. We also would like to express our thanks to Toon Calders and Bart Goethals for providing the source code allowing the extraction of frequent non-derivable patterns, Alain Casali and Stéphane Lopes for the source code allowing the extraction of the concise representation based on frequent essential patterns, and, Juho Muhonen and Hannu Toivonen for the source code allowing the extraction of frequent closed non-derivable patterns. This work is partly supported by the French-Tunisian CMCU project 05G1412.

References

1. Pasquier, N., Bastide, Y., Taouil, R., Stumme, G., Lakhal, L.: Generating a condensed representation for association rules. Journal of Intelligent Information Systems, Kluwer Academic Publisher, **volume 24(1)** (2005) 25–60
2. Calders, T., Goethals, B.: Non-derivable itemset mining. Data Mining and Knowledge Discovery (DMKD), Springer, **volume 14(1)** (2007) 171–206
3. Muhonen, J., Toivonen, H.: Closed non-derivable itemsets. In: Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2006), Springer-Verlag, LNAI, volume 4212, Berlin, Germany. (2006) 601–608
4. Casali, A., Cicchetti, R., Lakhal, L.: Essential patterns: A perfect cover of frequent patterns. In: Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2005), Springer-Verlag, LNCS, volume 3589, Copenhagen, Denmark. (2005) 428–437
5. Galambos, J., Simonelli, I.: Bonferroni-type inequalities with applications. Springer (2000)
6. Hamrouni, T., Ben Yahia, S., Mephu Nguifo, E.: Succinct minimal generators: Theoretical foundations and applications. To appear in the International Journal of Foundations of Computer Science (IJFCS). (2007)
7. Ganter, B., Wille, R.: Formal Concept Analysis. Springer (1999)
8. Hamrouni, T., Ben Yahia, S., Mephu Nguifo, E.: A new exact concise representation based on disjunctive closure. In: Proceedings of the 2nd Jordanian International Conference on Computer Science and Engineering (JICCSE 2006), Al-Balqa, Jordan. (2006) 361–373
9. Mielikäinen, T., Panov, P., Dzeroski, S.: Itemset support queries using frequent itemsets and their condensed representations. In: Proceedings of the 9th International Conference Discovery Science (DS 2006), Springer-Verlag, LNCS, volume 4265, Barcelona, Spain. (2006) 161–172
10. Ye, X., Keane, J.A.: Mining composite items in association rules. In: Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics (SMC 1997), Hyatt Orlando, Orlando, Florida, USA. (1997) 1367–1372
11. Zhao, L., Zaki, M.J., Ramakrishnan, N.: BLOSOM: A framework for mining arbitrary Boolean expression. In: Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining (KDD 2006), Philadelphia, PA, USA. (2006) 827–832
12. Calders, T., Rigotti, C., Boulicaut, J.F.: A survey on condensed representations for frequent sets. In: Constraint Based Mining and Inductive Databases, Springer-Verlag, LNAI, volume 3848. (2005) 64–80
13. Nanavati, A.A., Chitrapura, K.P., Joshi, S., Krishnapuram, R.: Mining generalised disjunctive association rules. In: Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001), Atlanta, Georgia, USA. (2001) 482–489
14. Kim, H.D.: Complementary occurrence and disjunctive rules for market basket analysis in data mining. In: Proceedings of the 2nd IASTED International Conference Information and Knowledge Sharing (IKS 2003), Scottsdale, AZ, USA. (2003) 155–157
15. Grün, G.A.: New forms of association rules. Technical Report TR 1998-15, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada (1998)

An FDP-Algorithm for Drawing Lattices

Christian Zschalig

Institut für Algebra, TU Dresden, Germany
`Christian.Zschalig@tu-dresden.de`

Abstract. In this work we want to discuss an algorithm for drawing line diagrams of lattices based on force directed placement (FDP). This widely used technique in graph drawing introduces forces acting on nodes and lines. A balanced state of the system will result in a diagram fulfilling the desired properties due to the underlying physical model. In our framework the aim was to maximize the conflict distance. In contrast to existing programs our approach provides attribute additive diagrams since forces act on $\bigwedge$-irreducibles only. Another relevant aspect is a careful initialization that helps to minimize the number of edge crossings.

1 Motivation

We observe a growing demand on visualizations of concept lattices for representing knowledge in FCA. Several programs [1, 7, 15] use diagrams for exploring and analyzing database structures. Unfortunately, the automatic layout of the diagrams remains a difficult task. In order to gain the acceptance of the user, who is in general not an expert, the drawings should be easily readable. However, nobody knows exactly what that means since it is obviously not possible to mathematize human esthetic sensations. Nevertheless, there exist some algorithms for drawing lattices with the computer [1, 9, 19] based on different approaches. In this work we want to present such an attempt combining the FCA view on diagram layouting with tools developed in the theory of graph drawing.

2 Preliminaries

2.1 How to Draw a Diagram

The *graph drawing* community developed a variety of methods to classify layout algorithms [5]. First we have the *drawing conventions* declaring general constraints of the resulting drawings. In our attempt this includes the following:

1. *line diagrams*, a common principle for drawing lattices automatically,
2. *upward diagrams*, sometimes also called Hasse-diagram, a common method in order to avoid arrows on diagram lines,
3. *attribute additive diagrams*, a principle introduced more generally in [11].

While the first two conventions are canonical in lattice drawing algorithms, the third is not. Also *layer diagrams* [9] or *hybrid diagrams* [2] are employed. The first alternative seems less applicable for our purpose since in general it does not emphasize the grid structure of a lattice. The second however offers some advantages in case of non-distributive lattices[1]. Nevertheless it still adheres to the layer convention.

The second technique to create algorithms is the employment of *esthetic criteria*. They are mostly given by optimization tasks whose compliance is supposed to increase the diagrams quality, i.e. readability. Examples include

– minimize the number of slopes,
– maximize the smallest angle between incident lines,
– put the nodes onto an orthogonal grid.

There is little empirical analysis about the importance of that criteria. Two studies were made [16, 17] but only for general lattices. They highlighted the criterion of *minimizing the number of edge crossings*. Therefore we tried to emphasize it in our algorithm. The second criterion taken into our consideration is *maximizing the conflict distance* [12], i.e. the least distance between a node and a non-incident line.

Finally, diagram algorithms are distinguished in classes characterizing the way how the actual layout process is done. The *layer method* (see [5] for an overview) is fairly prominent [2] as well as *force directed methods* (introduced in [8], see [5] for an overview) implemented in [19] or a combination of both [9]. We decided to implement a force directed method since it is a natural way to maximize the conflict distance. Unlike other attempts, we keep the diagram attribute additive, thereby better satisfying the esthetic criterion of *displaying symmetries*.

2.2 Diagrams of Lattices

We consider diagrams in the usual way (see for instance [14] for a formal definition). Instead of the lattice $\underline{\mathfrak{V}} = (\mathfrak{V}, \leq)$ itself we draw only its graph $(\mathfrak{V}, \prec)$ (where $\prec$ denotes the upward neighbourhood relation in $\underline{\mathfrak{V}}$). A *line diagram* (briefly *diagram*) is an injective mapping pos fulfilling the upward drawing convention that assigns a point $\mathrm{pos}(v)$ in the Euclidian plane (called node) to each lattice element v. An element $e = (v, w)$ of $\prec$ is mapped to a straight line segment between $\mathrm{pos}(v)$ and $\mathrm{pos}(w)$, for convenience we write $\mathrm{pos}(e)$ for the image.

Next we want to remind the already mentioned attribute additivity [11, 18]. Since we consider lattices instead of concept lattices in this work, we sloppily define: A diagram of a lattice $\underline{\mathfrak{V}}$ is *attribute additive* if the $\bigwedge$-irreducibles $m \in M(\underline{\mathfrak{V}})$ are assigned to vectors $\mathrm{vec}(m)$ and all elements v are mapped to the sum of the vectors of $\bigwedge$-irreducibles greater or equal than them, i.e.

$$\mathrm{pos}(v) = \sum_{m \geq v} \mathrm{vec}(m).$$

[1] That can be easily observed at the lattice M_n. Hybrid diagrams do not push the 0-element of the lattice disproportional downwards.

Finally we introduce the *conflict distance* due to [12]:

Definition 1 *Let $G = (V, E)$ be a simple graph and* $\mathrm{pos}(G)$ *a diagram of it. Let $v \in V$ be a vertex and $e \in E$ be a non-incident line w.r.t. v. The* node-line-distance *between* $\mathrm{pos}(v)$ *and* $\mathrm{pos}(e)$ *is the least Euclidian distance between* $\mathrm{pos}(v)$ *and any point $w \in \mathrm{pos}(e)$. The* conflict distance *of* $\mathrm{pos}(G)$ *is the smallest of all node-line-distances in* $\mathrm{pos}(G)$.

2.3 Left-Relations on Lattices

Left-relations give a possibility to characterize planar lattices and to describe plane diagrams of them [20–22]. They are closely related to *conjugate orders* [6]. Intuitively, a left-relation on a diagram describes, whether a lattice element v is left or right of another element w incomparable to v. In contrary, comparable elements are considered to be above or below each other. We do not need a formal definiton here but only one result given in [21] which explains the heuristics we use in Section 3.1, namely the *first planarity condition (FPC)*. In Figure 1 an intuitive explanation for the necessity of the FPC is given.

Definition 2 *[21] A conjugate relation*[2] *R on a lattice $\underline{\mathfrak{V}}$ fulfills the* first planarity condition (FPC) *if the implication $m_i \, R \, m_k \, R \, m_j \implies m_k > (m_i \wedge m_j)$ holds for all $\bigwedge$-irreducibles $m_i, m_k, m_j \in M(\underline{\mathfrak{V}})$.*

Proposition 1 *[21] Let L be a left-relation on a lattice $\underline{\mathfrak{V}}$, then the following equivalence holds:*

$$L \text{ satisfies the FPC} \iff \underline{\mathfrak{V}} \text{ is planar.}$$

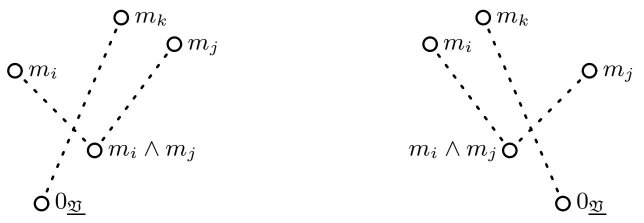


Fig. 1. When considering a diagram of a lattice, the necessity of the FPC is obvious for its planarity: If $m_i \, L \, m_k \, L \, m_j$ or $m_j \, L \, m_k \, L \, m_i$ holds then also $m_k > (m_i \wedge m_j)$. Otherwise every chain of diagram edges from m_k to the bottom element of the lattice intersects with a chain of edges from either m_i or m_j to $m_i \wedge m_j$.

[2] That is a relation satisfying $R \cup R^{-1} = \|$, where $\|$ denotes the incomparability relation in $\underline{\mathfrak{V}}$.

3 The FDP-Algorithm

Force Directed Placement (FDP) is a widely used technique for drawing diagrams of graphs [8, 10, 13] and is also applied for the layout of lattice diagrams [19, 9]. Nodes and sometimes also edges are considered as physical bodies which interact by a set of repulsive and attractive forces. If the system is in a balanced state, i.e. a local minimum of the appropriate energies, then the resulting diagram will hopefully be nicely drawn w.r.t. the properties that determine the forces. In general an FDP-algorithm consists of three parts:

– choice of an adequate model,
– determination of an initial state,
– seeking for a balanced state using an optimization algorithm.

In many cases attention is rather put on the optimization process, for instance with simulated annealing or genetic algorithms. The models are kept simple and the initial state is rather arbitrary. The two layout styles provided by [19] distribute the lattice elements on layers. This may work well if either a global minimum of the included energies is found or if several minima are found and ranked by a quality function acting on the respective diagrams.

Our approach tries to include a more sophisticated initialization. This is done by applying a heuristics to minimize the number κ of edge crossings in the diagram. The final optimization step is not allowed to shift any node out of its surrounding cell. Therefore, κ does not change as well as the left-relation of the diagram.

3.1 Initialization

As mentioned already this is the crucial step of our approach. Since the diagram is determined just by the coordinates of the $\bigwedge$-irreducibles we only have to assign vectors to them. We distinguish between the coatoms and other $\bigwedge$-irreducible elements. The first are distributed on a parable

$$y = -0.09x^2 - 1.75$$

which was derived heuristically from the position of the $\bigwedge$-irreducibles in suitable diagrams of the boolean lattices B_4 and B_5. In case of an even number of coatoms they obtain the coordinates given by the x-component $\pm 0.9, \pm 2.7, \ldots$, if otherwise an odd number occurs, we assign $0, \pm 1.8, \pm 3.6, \ldots$. All other vectors are obtained by

$$\text{pos}(m_i) = \Delta_i + \sum_{m_j > m_i} \text{pos}(m_j),$$

i.e. the mean of the vectors of $\bigwedge$-irreducibles m_j situated above m_i. If the $\bigwedge$-irreducible elements above m_i form a chain then the respective coordinates will lay on a line in the initial diagram. The symbol Δ_i represents a small shift which is necessary when two $\bigwedge$-irreducibles share the same upper neighbour.[3]

[3] Otherwise they would obtain the same diagram vector.

Finally we have to clarify in which way the nodes of the $\bigwedge$-irreducibles are sorted. We mentioned already that we want to minimize the number of edge crossings in the diagram. Since there exists no efficient analytical algorithm for that purpose we introduce a heuristical model called *planarity enhancer*.

The underlying idea is the following: *Similar concepts, i.e. concepts sharing similar intents should be positioned more closely than non-similar ones.* This is motivated by the FPC, see Proposition 1. It refers to the fact that, in a plane diagram, a $\bigwedge$-irreducible n should be drawn "inbetween" m_1 and m_2 only if it is greater than the infimum $m_1 \wedge m_2$, i.e. if $m_1 \wedge n \geq m_1 \wedge m_2$ or $m_2 \wedge n \geq m_1 \wedge m_2$. Hence pairs (m, n) of elements with a large infimum, i.e. with a small cardinality of $M(m \wedge n)$[4] shall be drawn close together. In order not to favor those $\bigwedge$-irreducible elements situated near the bottom of the lattice we only count $\bigwedge$-irreducibles not included in $M(m \vee n)$. Based on these ideas, we define the *sup-inf-distance* between two incomparable $\bigwedge$-irreducibles m and n as follows:

$$d_{SI}(m,n) := |M(m \wedge n) \setminus M(m \vee n)| - 1.$$

This allows us to create a complete weighted graph Γ_{SI} where $M(\underline{\mathfrak{V}})$ is the vertex set and each edge between m and n obtains the weight $d_{SI}(m,n)$. This graph can be considered as a 2-dimensional physical body with rings instead of nodes and springs instead of edges. The spring force is due to the standard physical model given as $F = -k \cdot x$, where k is the spring constant and x the displacement of the idle state. The springs are thought to be in rest position if their length is equal to the weight in the appropriate graph. This results in a system energy

$$E_{SI} = \sum_{m_i, m_j \in M(\underline{\mathfrak{V}})} (|\mathrm{pos}(m_i) - \mathrm{pos}(m_j)| - d_{SI}(m_i, m_j))^2.$$

We find the force acting on a vector $\mathrm{pos}(m)$ by differentiating the last formula to each $\bigwedge$-irreducible m yielding

$$F_{SI}(\mathrm{pos}(m)) = -2 \cdot \sum_{n \in M} \frac{|\mathrm{pos}(m) - \mathrm{pos}(n)| - d_{SI}(m,n)}{|\mathrm{pos}(m) - \mathrm{pos}(n)|} \cdot (\mathrm{pos}(m) - \mathrm{pos}(n)).$$

After reaching an equilibrium state of this system by applying a robust minimizer we do a linear regression of the emerged scatter plot, followed by an orthogonal projection of the nodes onto the obtained line. The sorting of the nodes representing the attributes finally gives their sorting relation [20]. This is a relation indicating in which order the $\bigwedge$-irreducibles with common upper neighbour shall be sorted from left to right.

[4] With $M(v)$ we denote the set of $\bigwedge$-irreducibles greater or equal than v.

3.2 The Model

Implemented Forces

Since we want to draw attribute additive diagrams the energies and forces do not act on the node of an element $v \in \mathfrak{V}$ itself but on the vectors $\mathrm{vec}(m)$ of the $\bigwedge$-irreducibles $m \in M(v)$. Therefore, the systems energy is the sum of the ones inherent in the vectors of the elements of $M(\underline{\mathfrak{V}})$. Equivalently, the resulting force is a vector of forces on vectors on $\bigwedge$-irreducibles (see Figure 2). This yields

$$E = \sum_{m \in M} E(\mathrm{vec}(m_i)) \text{ and } F = -\nabla E = (F(\mathrm{vec}(m_1)), \ldots F(\mathrm{vec}(m_0)))^T.$$

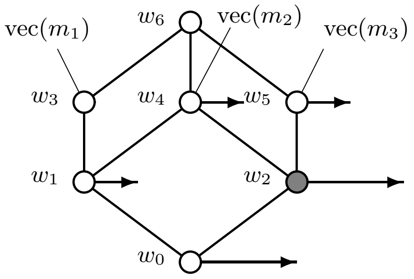


Fig. 2. The force acting on the node w_2 affects the nodes w_0, w_1, w_4 and w_5 too. Since the latter three contain only one of the $\bigwedge$-irreducibles m_2 or m_3, the resulting force is half the original.

The aim of the model is to maximize the conflict distance. It is therefore based on a *repulsive force* F_{rep}. Since we want to avoid the occurence of any conflict, the following definition on the graph $(\mathfrak{V}, \prec)$ of a lattice is obvious:

$$F_{rep} = -\nabla E_{rep}, \quad E_{rep} = \sum_{v \in V} \sum_{e \in \prec, v \notin e} \frac{1}{d(\mathrm{pos}(v), \mathrm{pos}(e))}.$$

Thereby $d(\mathrm{pos}(v), \mathrm{pos}(e))$ is the distance between a node and an edge introduced in Definition 1. A node positioned on a non-incident line causes E_{rep} to be infinite, hence this never results in a stable state.

To prevent the diagram from blowing up we need an *attractive force* F_{att} which minimizes the edge length. This is done due to the physical model of a spring supplying the formulas

$$F_{att} = -\nabla E_{att}, \quad E_{att} = \sum_{e \in \prec} |\mathrm{pos}(e)|^2.$$

Finally, we employ a *gravitative force* F_{grav} to ensure the upward-drawing-constraint. It acts on vectors $n_i = \mathrm{vec}(m_i)$ of $\bigwedge$-irreducibles only. Since this force is supposed to be dependent on the angle $\varphi(n)$ between the vector of an $\bigwedge$-irreducible and a horizontal line, we define as follows:

$$F_{grav}(n_i) = -\frac{\mathrm{d}E_{grav}}{\mathrm{d}n_i},$$
$$\frac{\mathrm{d}E_{grav}(n_i)}{\mathrm{d}\varphi(n_i)} = \frac{\sin^2\varphi(n_i) - \sin^2\varphi_0}{\sin^2\varphi(n_i)} \cdot \begin{cases} 1 \,, \varphi(n_i) \in [0, \varphi_0] \\ 0 \,, \varphi(n_i) \in [\varphi_0, \pi - \varphi_0] \\ -1 \,, \varphi(n_i) \in [\pi - \varphi_0, \pi] \end{cases}$$

This formula seems to be quite clumsy. However, the underlying idea is to push "nearly horizontal" vectors stronger downwards than the more slanted ones (see Figure 3). The gravitative force may not act if the vectors are "vertical enough", i.e. if their angle is between φ_0 and $\pi - \varphi_0$, which is chosen by (see Figure 3 for an explanation)

$$\varphi_0 := \frac{\pi}{|M(\underline{\mathfrak{B}}(\mathbb{K}))| + 1} \,.$$

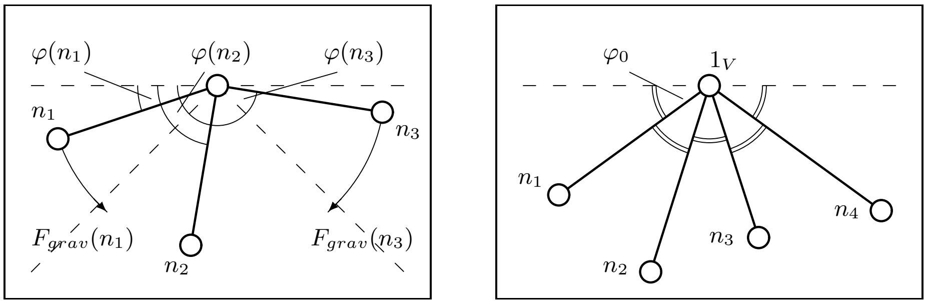


Fig. 3. left: The diagonal lines represent φ_0 and $\pi - \varphi_0$. The nodes n_1 and n_3 are pushed down by the gravitative force, but not n_2.
right: If all attribute concepts are coatoms then their vectors can be assigned, s.t. the angles between two of them and to the horizontal dashed line are all equal to φ_0.

Integrating by φ_0 in consideration of the reasonable boundary condition given by $\varphi(n_i) \in \{\varphi_0, \pi - \varphi_0\} \implies E(n_i) = 0$ to make the energy function continuous in $(0, \pi)$ yields

$$E(n_i) = \begin{cases} \varphi(n_i) + \cot\varphi(n_i)\sin^2\varphi_0 + E_0 \,, 0 \le \varphi(n_i) \le \varphi_0 \\ -\varphi(n_i) - \cot\varphi(n_i)\sin^2\varphi_0 + E_1 \,, \pi - \varphi_0 \le \varphi(n_i) \le \pi \end{cases}$$
$$\text{with } E_0 = E_1 - \pi = -\varphi_0 - \sin\varphi_0\cos\varphi_0 .$$

The total energy and the total force respectively are obtained as a linear combination of its components, i.e.

$$E = r \cdot E_{rep} + a \cdot E_{att} + g \cdot E_{grav},$$
$$F = r \cdot F_{rep} + a \cdot F_{att} + g \cdot F_{grav}.$$

Calculation of the Forces

The calculation of the repulsive force demands the observation of various cases. Firstly we have to distinguish how a node and a non-incident line are related to each other in the plane. We discover the three possibilities given in Figure 4.

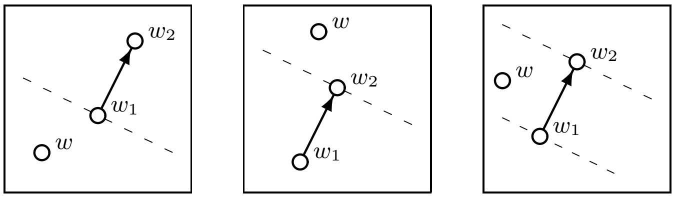


Fig. 4. The three cases of node-edge-relationship.

Secondly we must take into consideration which of the sets $M(v)$, $M(v_1)$ and $M(v_2)$ contain the $\bigwedge$-irreducible m. The alternatives are depicted in Figure 5.

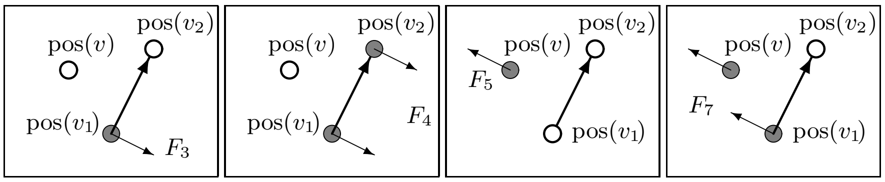


Fig. 5. Four possibilities of forces acting between a node and a non-incident line depending on the containment of a $\bigwedge$-irreducible m in the sets $M(v)$, $M(v_1)$ and $M(v_2)$ of the respective lattice elements (shaded).

After some calculation we find the following table determining the repulsive force for each case. Due to abbreviation reasons we write n instead of pos(m) for attribute vectors, w instead of pos(v) for nodes and f instead of pos(e) for vectors of diagram lines. The symbol e_w denotes the unit vector of a node w, furthermore $n_+(z)$ is the vector arising from the vector z by turning by $\pi/2$ in positive direction of rotation and l either $+1$ in case of the node w being situated left of the line f and -1 otherwise.

	F_3	F_4	F_5	F_7
case 1	$e_{(w_1-w)}$	$e_{(w_1-w)}$	$e_{(w-w_1)}$	0
case 2	0	$e_{(w_2-w)}$	$e_{(w-w_2)}$	$e_{(w-w_1)}$
case 3	$-\sqrt{\frac{(w_2-w)^2-\|h\|^2}{\|f\|^2}} \cdot \frac{n_+(f)\cdot l}{\|f\|}$	$-\frac{n_+(f)\cdot l}{\|f\|}$	$\frac{n_+(f)\cdot l}{\|f\|}$	$\sqrt{\frac{(w_1-w)^2-\|h\|^2}{\|f\|^2}} \cdot \frac{n_+(f)\cdot l}{\|f\|}$

Table 1. Summary of all occuring forces $\frac{\mathrm{d}d(w,f)}{\mathrm{d}n_i}$ for the different cases.

The appropriate repulsive force acting on an attribute vector n_i is given by

$$F_{rep}(n_i) = -\frac{\mathrm{d}E_{rep}(n_i)}{\mathrm{d}n_i} = -\sum_{v\in V}\sum_{e\in\prec, v\notin e}\frac{\mathrm{d}\left(\frac{1}{d(w,f)}\right)}{\mathrm{d}n_i}$$
$$= \sum_{v\in V}\sum_{e\in\prec, v\notin e}\frac{1}{d(w,f)^2}\cdot\frac{\mathrm{d}d(w,f)}{\mathrm{d}n_i}.$$

For the attractive force we gain the formula

$$F_{att}(n_i) = -\sum_{e\in\prec}\frac{\mathrm{d}|f|^2}{\mathrm{d}n_i} = \quad 2\cdot\sum_{\substack{v_1v_2\,\in\prec,\\ m_i\,\in\, v_1\setminus v_2}} f,$$

and finally for the gravitative force

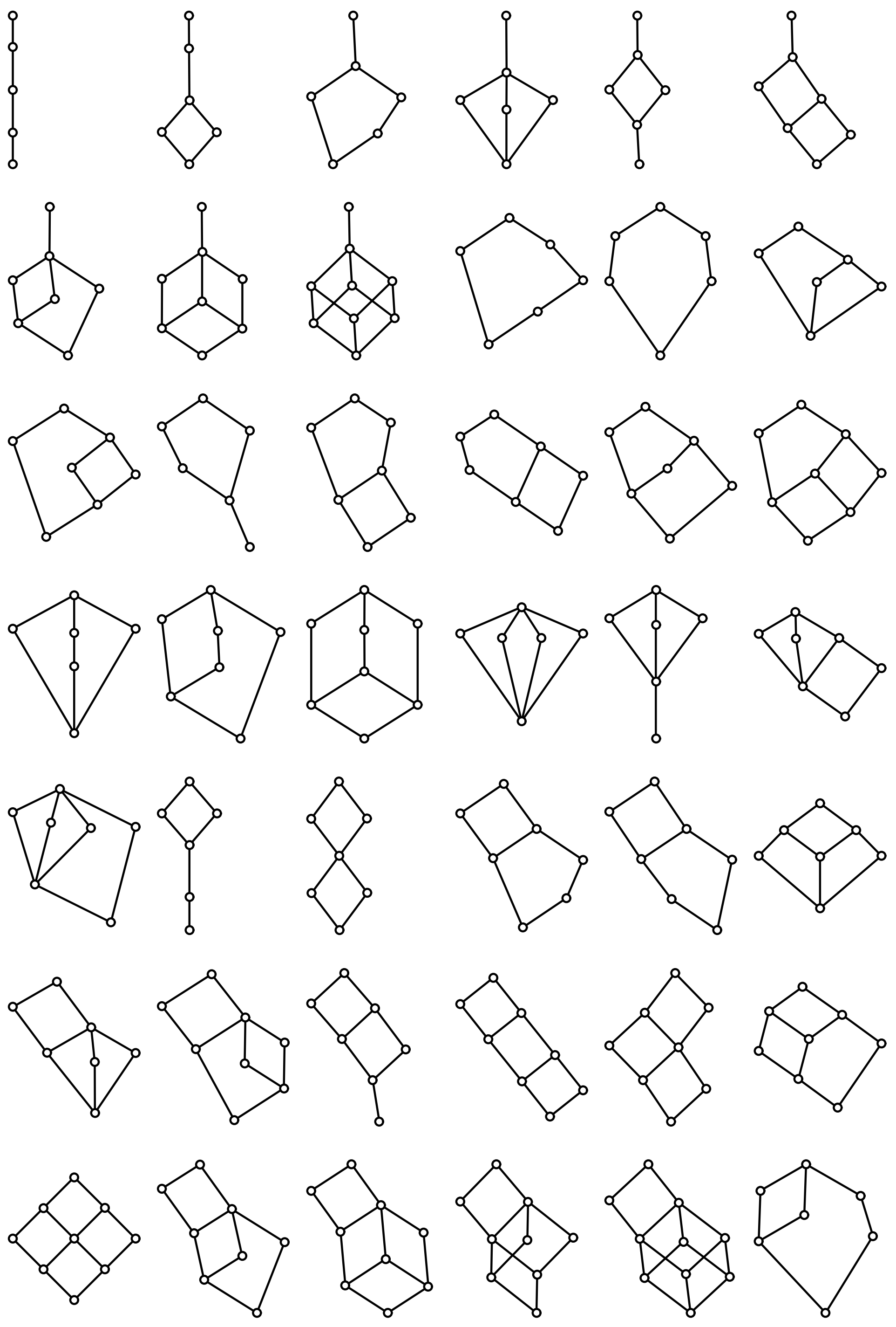
$$F_{grav}(n_i) = -\frac{\mathrm{d}E_{grav}(n_i)}{\mathrm{d}n_i} = n_+(n_i)\cdot\frac{\sin^2\phi(n_i) - \sin^2\phi_0}{y(n_i)^2}\cdot\begin{cases} 1, & 0 < \phi(n_i) < \phi_0 \\ -1, & \phi_0 < \phi(n_i) < \pi \end{cases}.$$

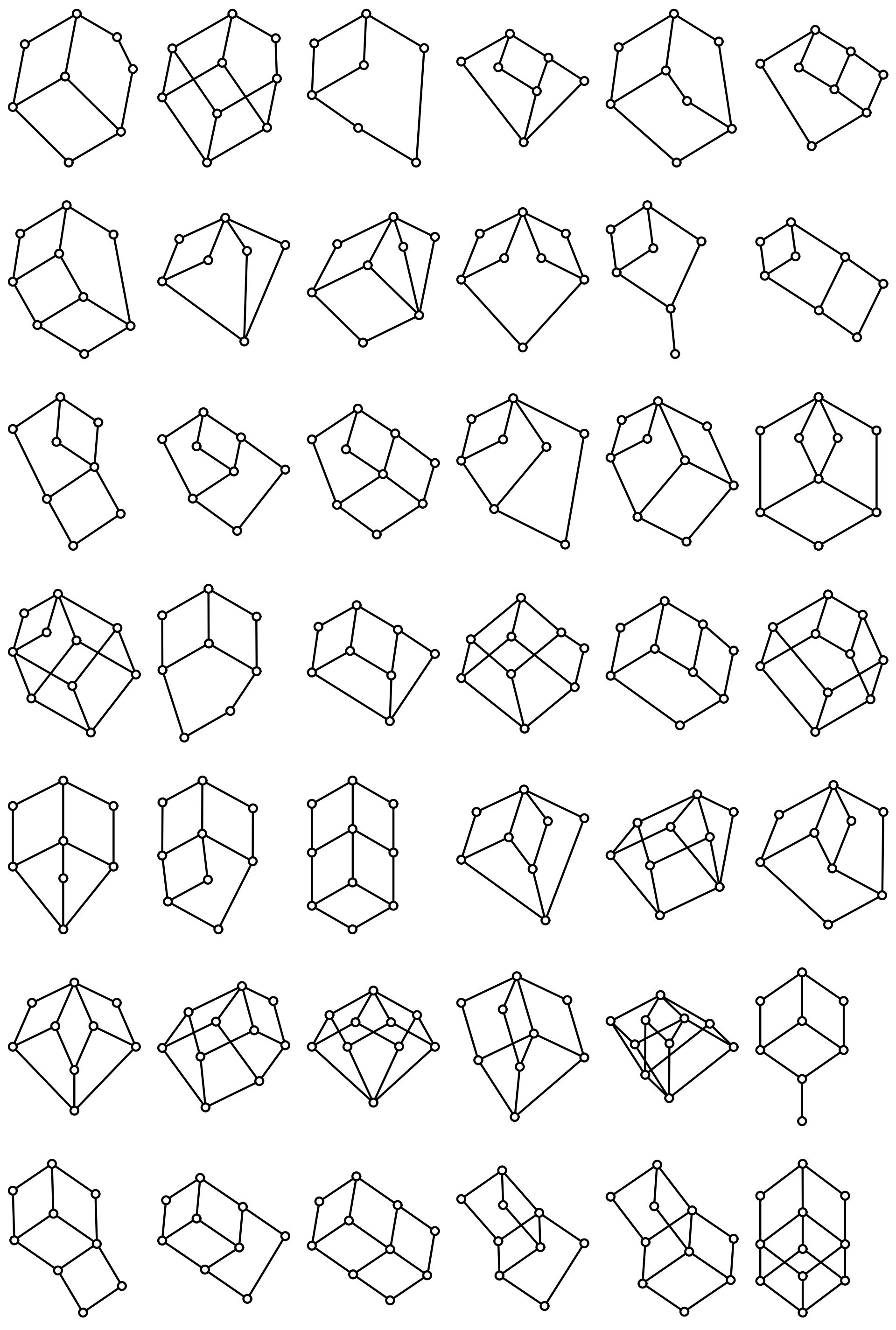
3.3 The Minimizing Algorithm

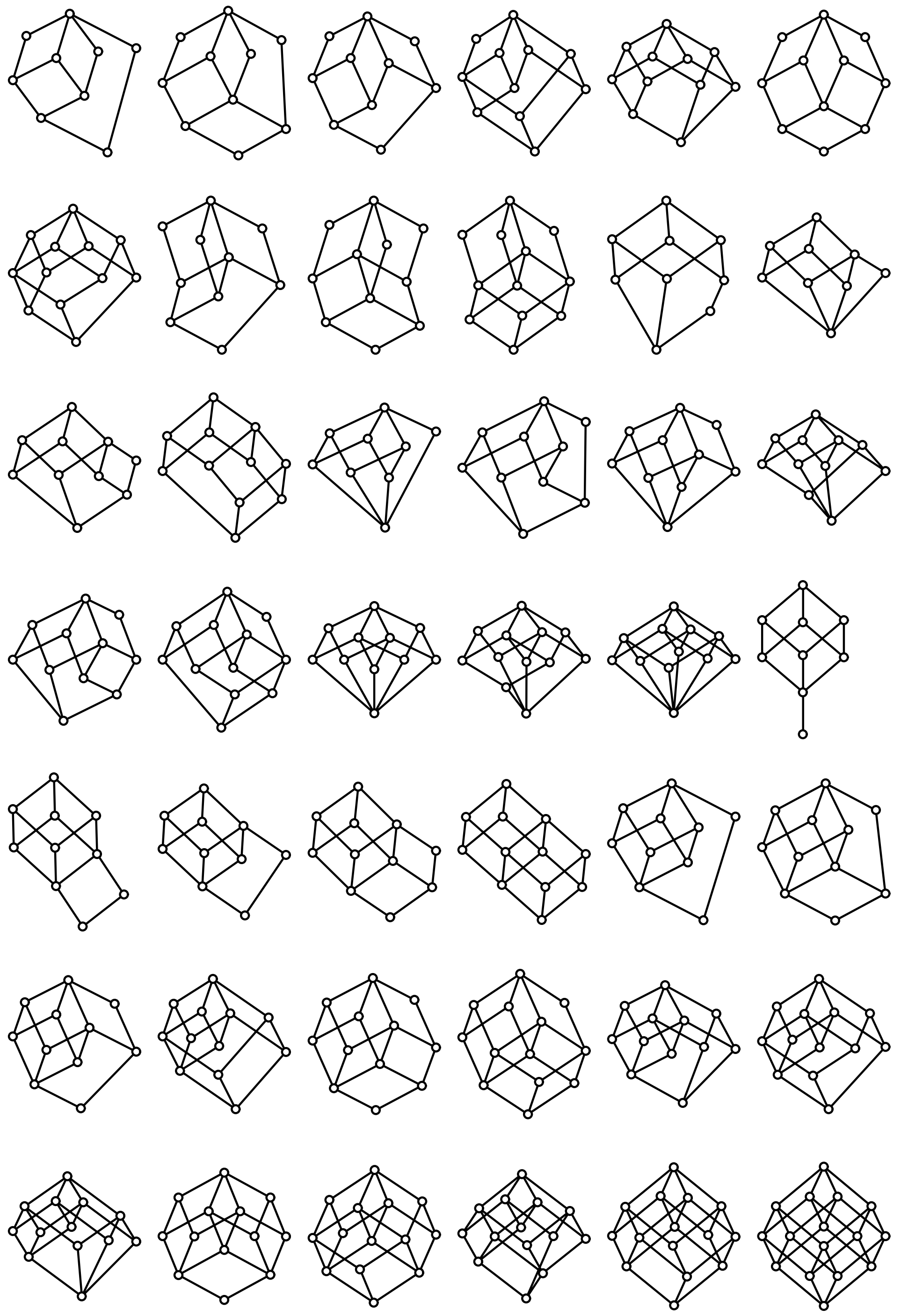
This step requires robustness rather than speed. We definitely want to avoid a node stepping out of its cell in order to keep the shape of the initialized diagram. Running time is no issue, the calculation of the local minimum is done in less than a second in most cases. We chose the well known conjugate gradient method as optimization algorithm (see for instance [4]).

4 Results

Instead of displaying some arbitrary chosen diagrams we just give the drawings of all lattices with four $\bigwedge$-irreducible elements. The calculation of all diagrams took less than 10 seconds. We also produced the respective diagrams of lattices with five $\bigwedge$-irreducible elements. The process of determining all 13596 lattices and drawing their diagrams took 25 minutes on an ordinary PC, which is an average of about 0.1 seconds per diagram. Unfortunately they are not publishable due to space limitations.







5 Conclusion

We presented an algorithm for drawing lattice diagrams designed in three steps. The *initialization* serves for implementing some esthetic criteria whose compliance is desired for the resulting drawing. In our particular approach we chose *minimization of the number of edge crossings* as such an criterion. The second step, namely the choice of a *model*, aims to *maximize the conflict distance*, thereby considering particular *drawing conventions*. Here we chose the *attribute additivity*. Finally, the optimization step seeks for a minimal energy state (i.e. a maximal conflict distance) while keeping the properties of the initial state and the model. Ino ur algorithm the underlying left-relation and the attribute additivity are preserved.

We think that our approach is fairly convenient for drawing lattice diagrams. The framework of an initialization employing esthetic criteria and an FDP-algorithm including drawing conventions gives a clear image of the desired properties of the resulting diagram. This modularity seems to be the main advantage of this approach.

As already mentioned, diagrams with few edge crossings are favoured by users. Therefore a drawing algorithm should emphasize that issue. Although our approach contains a very simple initialization heuristics only, all planar lattices with four $\bigwedge$-irreducibles are indeed drawn without edge crossings.

Layer assignment is a favoured way of lattice drawing algorithms. Even though we think that layer diagrams are satisfactory and user preferred for many (in particular distributive) lattices, one should not restrict to this convention in general.

Despite these advantages of our approach we do not think that the results are superior to those of other algorithms. In the following we want to give some possibilities that could improve the diagrams quality:

- The involvement of additional or different constraints like the *visualization of chains* (by chain decomposition) or the *hybrid convention* could result in more symmetrical diagrams.
- The initialization step can be improved by including techniques given in [20–22] to improve the quality of layouts of planar lattices. Also, we recently try to find strategies to characterize and draw “nearly planar” lattices as well.
- It may be useful to produce several diagrams that can be compared by some set of quality functions (proposed for instance in [3]).

References

1. P. Becker, J. Hereth, G. Stumme, *ToscanaJ - an open source tool for qualitative data analysis.* FCAKDD 2002, pp. 1-2, 2002.

2. R. Cole, *Automated Layout of Concept Lattice Using Layer Diagrams and Additive Diagrams.* Austr. Comp. Sc. Conf., 2000.
3. R.J. Cole, J. Ducrou, P. Eklund, *Automated Layout of Small Lattices Using Layer Diagrams.* Proc. of ICFCA06, LNAI 3874, pp. 291-305, 2006.
4. P. Deuflhard, A. Hohmann, *Numerische Mathematik - Eine algorithmisch orientierte Einführung.* de Gruyter Verlag, 1991.
5. G. DiBattista, P. Eades, R. Tamassia, I. G. Tollis, *Graph Drawing.* Prentice Hall, 1999.
6. B. Dushnik, E.W. Miller: *Partially Ordered Sets.* Amer. J. Math. 63, 1941, pp. 600-610.
7. J. Ducrou, P. Eklund, *Combining Spatial and Lattice-Based Information Landscapes.* Proc. of ICFCA05, LNAI 3403, pp. 64-78, 2005.
8. P. Eades: *A Heuristic for Graph Drawing.* Congressus Numerantium 42, pp. 149-160, 1984.
9. R. Freese, *Online Java Lattice Building Application.* `http://maarten.janssenweb.net/jalaba/JaLaBA.pl`
10. T. Fruchterman, E. Reingold, *Graph Drawing by Force Directed Placement.* Software - Practice and Experience 21, no.11, pp. 1129-1164, 1991.
11. B. Ganter, R. Wille: *Formal Concept Analysis.* Springer, 1999.
12. B. Ganter: *Conflict Avoidance in Order Diagrams.* preprint, TU Dresden, 2003.
13. S. Kamada, T. Kawai, *An Algorithm for Drawing General Undirected Graphs.* Information Processing Letters 31 (1989), pp. 7-15, 1989.
14. D. Kelly, I. Rival: *Planar Lattices.* Can. J. Math. Vol. 27, No. 3, pp. 636-665, 1975.
15. B. Köster: *FooCA: Enhancing Google Information Research by Means of Formal Concept Analysis.* Contributions of ICFCA06, Verlag Allgemeine Wissenschaft, 2006.
16. H. Purchase, *Which Aesthetic Has the Greatest Effect on Human Understanding?* Symposium on Graph Drawing GD'97, LNCS 1353, pp. 248-261,1997.
17. H. Purchase, J. Adler, D. Carrington, *User Preference of Graph Layout Aesthetics: A UML Study.* Symposium on Graph Drawing GD'00. LNCS 1984, pp. 5-18, 2000.
18. M. Skorsky: Endliche Verbände - Diagramme und Eigenschaften. Dissertation, TH Darmstadt, 1992.
19. S. Yewtushenko et al., *Conexp.* `http://conexp.sourceforge.net`
20. C. Zschalig: *Planarity of Lattices - An approach based on attribute additivity.* Proc. of ICFCA 05, LNAI 3403, pp. 391-402, 2005.
21. C. Zschalig: *Characterizing Planar Lattices Using Left-relations* Proc. of ICFCA06, LNAI 3874, pp. 280-290, 2006.
22. C. Zschalig: *Bipartite Ferrers-Graphs and Planar Concept Lattices.* Proc. of ICFCA07, LNAI 4390, pp. 313-327, 2007.

What can lattices do for math. teaching & education?

Vincent Duquenne

CNRS-ECP6, Université Pierre et Marie Curie,
175 rue du Chevaleret, 75013 Paris, France
duquenne@math.jussieu.fr

Abstract. A reasonable size example coming from cognitive psychology is re-analyzed with standard tools of FCA and Lattice Analysis. Developmental shifts / classifications are explored on a descriptive and graphical viewpoint, through attribute implications and unglued decomposition in regular intervals. This assesses child similarities in performance and behavior, while comparing intervals focuses on what children miss collectively to make further progress.

Keywords: (nested) line diagrams, (relative) canonical basis of implications, unglued decomposition, developmental shifts, construction of natural numbers.

Introduction

A first aim behind this paper is to address two topics of the conference *CLA'07* : "visualizing inherent properties in data sets" / "classifying systems based on relationships among objects and attributes through the concept of mathematical lattices" and to make this address concrete on a real example. In such a way the graphical outputs do not reduce to usual small models though staying still readable.

Secondly, to try convincing the researchers in educational / cognitive sciences that basic tools of Lattice Theory [Bi67] and Formal Concept Analysis (FCA, [GW99]) – which we usually mix together in French as *Analyse Latticielle* [D99]- can be most useful for describing their data and help in modeling underlying processes. All over the years, we have had many collaborations in applying Lattice Analysis to topics from the social and medical sciences (see [D99] and application papers quoted there) up to genetics (see [C&Al01], [Do&Al01], and [D&A01]), but not with that many cognitive scientists, although FCA has well spread in their close community of AI.

Third motivation, to try sharing our experience in visualizing lattice properties with our computer program (GLAD: General Lattice Analysis & Design, see [D83-96]). Instead of giving the program code which has been asked sometimes in conferences and surrounding communities -but would not make sense since it's now of an oldish system dependent conception- it seems to us far better to display convincing examples following carefully chosen features and to illustrate specifications through examples.

Last, as we did already twenty years ago with a paper attesting the usefulness of Lattice Theory and some of its standard tools to formalize and generate experimental designs (by characterizing them as partition sublattices see [D86]), it's a bow to the celebrated paper by G. Birkhoff: "What can lattices do for you?" ([Bi70]) which primed our own interest in applications of Lattice Theory to surrounding disciplines.

To follow these objectives, we will come back to a data set that Dr. Camilo Charron collected -and analyzed with statistical methods in his thesis (see [Cha98])-. This example can be classified between cognitive sciences / developmental & educational psychology. A small data subset has been extracted from a chapter devoted to "developmental shifts and knowledge transformations along the construction of natural and relative numbers" in children (4-14 years old).

Naturally, we will not enter deeply in the questions of semantics and interpretations, referring to the original work -or subsequent papers- to get more precise descriptions of the psychological and educational setting. We will follow a go-between attitude, linking separated topics and communities: to say enough about the content so that a user of FCA can understand the kind of specific questions encountered there, and reciprocally to illustrate and comment the tools and techniques with simple words so that psychologists can foresee the kind of drawbacks they could get after a small investment in papers on FCA giving them more abilities and insights.

Basic data and original questions

Out of the eight groups of children under study, only three are kept here (age 4.5 / 5.5 / 6.5 years old). They are -in French- denoted MS:"moyenne section", GS:"grande section" and CP:"cours préparatoire", the latter being the first year of elementary school, the others being the last two years of nursery school. Each group comprises 31 children, which passed a series of ten –"*à la Piaget*"- experiments to evaluate their mastering of operations and relations on natural numbers. Hence for each group of children the basic data consists in a binary table $C_{31}xA_{10}\rightarrow\{0,1\}$, for which (c,a)=1 whenever the child "c" masters attribute "a", and (c,a)=0 otherwise. The ten attributes kept here concern only properties of *natural numbers*, operations and relations: for instance *order-ct* means mastering: [a>b & c=d implies a+c > b+d].

A:order	**F:**difference
B:equality	**G:**class-equiv.
C:order-ct	**H:**counting
D:equality-ct	**I:**identity-c
E:addition	**J**:commutativ.

(ct:conservation by translation, c:conservation)

Table 1. The attributes describing properties and operations on *natural numbers.*

As claimed in the thesis "The aim is to detect shifts [*ruptures*] of development along the construction of natural and relatives numbers…", which is made precise later with additional hypothesis: "Child developments will be partially ordered, which will be assessed by child profiles [*patrons de réponses*] that will be genetically ordered and structured by exact [*as opposed to association rules*] implications. Implications will point out shift and / or knowledge transformations". The main questions will be taken in charge naturally by *implications* which are one of the basic tools of FCA ([GW99 §2.3]), and these cognitive questions have an *intensional* nature. We will try to show that other natural questions relative to child classifications can also be raised, addressing more *extensional* –and educational- questions.

Lattice and analysis toolkit

The prerequisites for mastering the analysis and understanding the graphics comprise a few standard theorems and properties of FCA (see [GW99]) and applications of lattice analysis ([D99]). They will not be repeated here in mathematical terms, that would be useless and redundant. However they will be commented in everyday words, as a reminder and so that not-lattice-minded-users can foresee the content.

The basic data -for each group- can be seen as a binary relation $I \subseteq C_{31} x A_{10}$which is represented by a so-called "*context*" $(C_{31}, A_{10}, I \subseteq C_{31} x A_{10})$ keeping tracks of the children and attribute labels, where $(c,a) \in I$ means children "c" masters attribute "a". For a subset $B \subseteq A_{10}$ of attributes let $B^{\downarrow} := \{c \in C_{31} / cIa \text{ all } a \in B\}$ be its *extension*. For a subset $D \subseteq C_{31}$ of children let $D^{\uparrow} := \{a \in A_{10} / dIa \text{ all } d \in D\}$ be its *intension*. As $B^{\downarrow\uparrow\downarrow} = B^{\downarrow}$ (all $B \subseteq A_{10}$) and $D^{\uparrow\downarrow\uparrow} = D^{\uparrow}$ (all $D \subseteq C_{31}$), the pair of maps $B \rightarrow B^{\downarrow}$ (all $B \subseteq A_{10}$) and $D \rightarrow D^{\uparrow}$ (all $D \subseteq C_{31}$) defines a *Galois connection* between the power sets of A_{10} and C_{31}, and two *closure operators* in them, that gives the matching extensions / intensions. In words, by the *Galois* / *concept lattice* construction the context is unfolded in a *concept system* which can be drawn, each *concept* being defined both in extension / intension, and the organization of concepts being driven upwards by the *join operation* (defined by intersection of intensions), and dually downwards by the *meet operation* (defined by intersection of child groups). These underlying mathematical structures are known since the first 1940 edition of [Bi67], and papers by O. Ore (see [O44]) and others.

Now a lattice can become really cumbersome and complex to draw, so that it is most often useful to *label* it *minimally* by locating each attribute $a \in A$ at $(a^{\downarrow}, a^{\downarrow\uparrow})$ i.e. the higher concept to which it belongs, and dually to locate each child $c \in C$ at $(c^{\uparrow\downarrow}, c^{\uparrow})$ that is the lowest concept having c in its extension. Pointing to any lattice element, its intension can be reconstructed by taking all attributes above it along the ordered lattice, and its extension comprises all children below it. A second drawing simplification introduced since the beginning of FCA [Wi82] is to start a lattice drawing for only a subset of attributes, and to introduce the remaining in a *nested line diagram* erasing lines parallel to those that are grafting these remaining attributes.

The duality between extensions / intensions is also carrying *implications* between attributes (symmetrically between children, that are meaningful in a *social network* perspective: who is together with whom?). When two attributes $a,b \in A$ are such that $a^{\downarrow} \subseteq b^{\downarrow}$ -so that $a<b$ in the lattice- this can be read as a *simple* (premise) implication $a \rightarrow b$. When $(ab)^{\downarrow\uparrow} = abcd$, this indicates that $ab \rightarrow cd$ holds in the data. Both kind of implications can be deciphered graphically in the lattice. The simple ones will define the (*pre*)*order of attributes*, while the latter will be recognized by: the meet of a and b will "capture" c,d upwards. Fortunately, there is a *canonical basis of implications* summarizing all those holding in the data, which was the main result of [GD84-86] (see also [D84-87] for a more latticial version and [G84-87] for a nice algorithm).

Another procedure that will be used is the lattice *unglued decomposition* [GW99 §4.2]. The original idea came from classes of lattices encountered in Mathematics -*distributive*, *modular*- since they are decomposable in maximal *atomistic* intervals, a construction that have been generalized to arbitrary lattices using *tolerances* (i.e. *similarities* respecting lattice operations as lattice *congruences* do) and the lattice *cover* relation. In words, it is a way to look at the lattice "from further" by considering faithful similarities between attributes, and symmetrically upwards between children.

Intra-group results and analysis

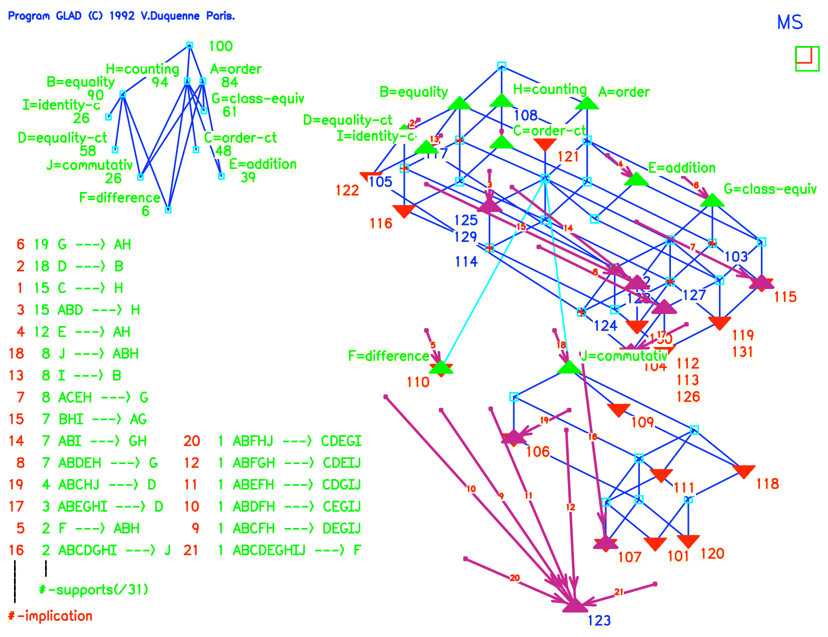
For the first group ("MS", average age 4.5 years) the order of attributes –weighed with the % of the attribute's extension / the 31 children- is displayed in Fig. 1 (top left). It should be noted that three attributes are somehow "easy" for that group since they are already mastered by almost all children (*counting* 94%, *equality* 90%, *order* 84%). Many other attributes imply some of the latter, creating an attribute order of *length* one. In particular, the "most difficult" attributes for this group of children (*identity-c*, *commutativ*. 26%, and *difference* 6%) imply all three easy attributes.

A nested line diagram of the lattice is unfold (top right), along *commutativ.* and *difference* -meaning that the lines parallel to the one joining *commutativ.* to its upper cover have not been drawn for simplifying the drawing. First remark completing the fact that there are many order relationships / dependence between pair of attributes, the lattice has 53 elements, which is small as compared to the 2^{10}=1024 subsets of A_{10}, that would be observed if there were a complete independence of the 10 attributes.

In the powerset of attributes this means that the closure operator $B|\rightarrow B^{\downarrow\uparrow}$ (all $B\subseteq A$) that generates the intensions, at the same time generates a lot of equivalence and implications between conjunctions of attributes. This can be summarized by the canonical basis of implications, which is listed at left hand side below the order of attributes in Fig. 1, together with the identification number 1-21 and the extensions / supports. The implications are also grafted into the nested line diagram of the lattice, where they can be located by their identification number and intension, which is obtained from the list by taking the union of their premise and conclusion.

Hence for instance, the first simple implication (n°-6) at the list top, is $G \rightarrow AH$, and means that the 19 children mastering G=*class-equiv.* master A=*order* & H=*counting* as well. It can be located at the top right of the lattice and also deciphered in the lattice structure by the fact that the intension of the element labelled G is GAH, since collecting G and H along the lattice lines above G. As for non-simple implication with complex premises, for instance below implication n°-6 is located implication n°-7 -which reads $ACEH \rightarrow G$- indicates that the eight children that master A=*order* & C=*order-ct* & E=*addition* & H=*counting* master *class-equiv.*
In the basis list, the implications have been ranked by extension's decreasing order. Within the lattice, closer their intension is to the lattice top, bigger their intension is obviously. Dually, closer to the bottom they have smaller extensions. Now some implications could be understood as natural / obvious / "by construction". Thus, that D=*eqality-ct* $\rightarrow$ *B=equality* will not surprise anybody, even if their percentage extensions 58% / 90% may require some comments. Other implications may come from sampling questions: after all 31 children for exploring and assessing something definite on a universe with 2^{10} elements is not a lot… Hence, the basis of implications should be scanned thoroughly by the researcher with all these considerations in mind.

On a "macro level", the lattice is decomposable in six intervals (see Fig. 2) by unglued decomposition, which is quite a strong since rare property for arbitrary lattices. These intervals are ordered along a 2x3 product of chains. This gives a macro scaling of attributes of "similar difficulty level" –regarding that group of children-, with the easy attributes *counting-equality-order* to which is now added *class-equiv.* at the top, which are followed downwards by more difficult attributes scaled in two



```
 6 19 G ---> AH
 2 18 D ---> B
 1 15 C ---> H
 3 15 ABD ---> H
 4 12 E ---> AH
18  8 J ---> ABH
13  8 I ---> B
 7  8 ACEH ---> G
15  7 BHI ---> AG
14  7 ABI ---> GH        20  1 ABFHJ ---> CDEGI
 8  7 ABDEH ---> G      12  1 ABFGH ---> CDEIJ
19  4 ABCHJ ---> D      11  1 ABEFH ---> CDGIJ
17  3 ABEGHI ---> D     10  1 ABDFH ---> CEGIJ
 5  2 F ---> ABH         9  1 ABCFH ---> DEGIJ
16  2 ABCDGHI ---> J    21  1 ABCDEGHIJ ---> F
|   |
|   #-supports(/31)
#-implication
```

Fig. 1. Attribute order (top left) and nested line diagram of the lattice (right) along *difference* & *commutativ*., with canonical basis of 21 implications and their supports.

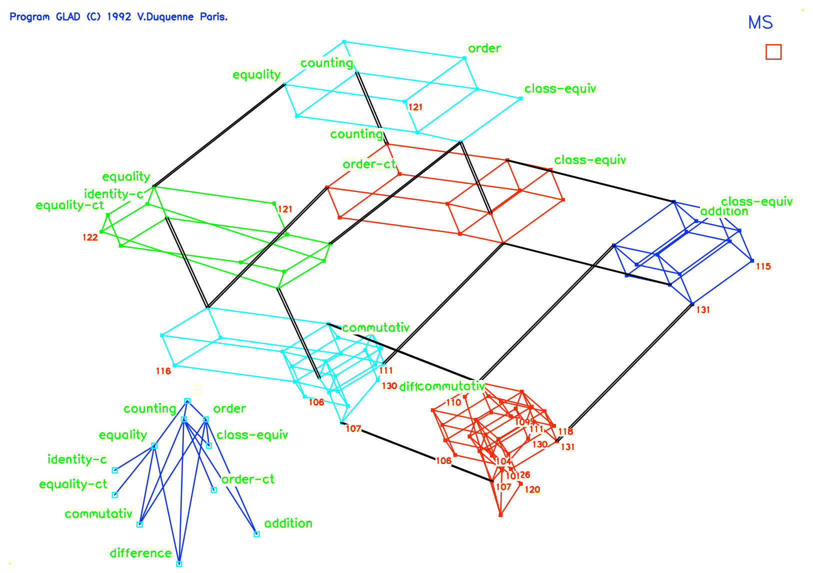


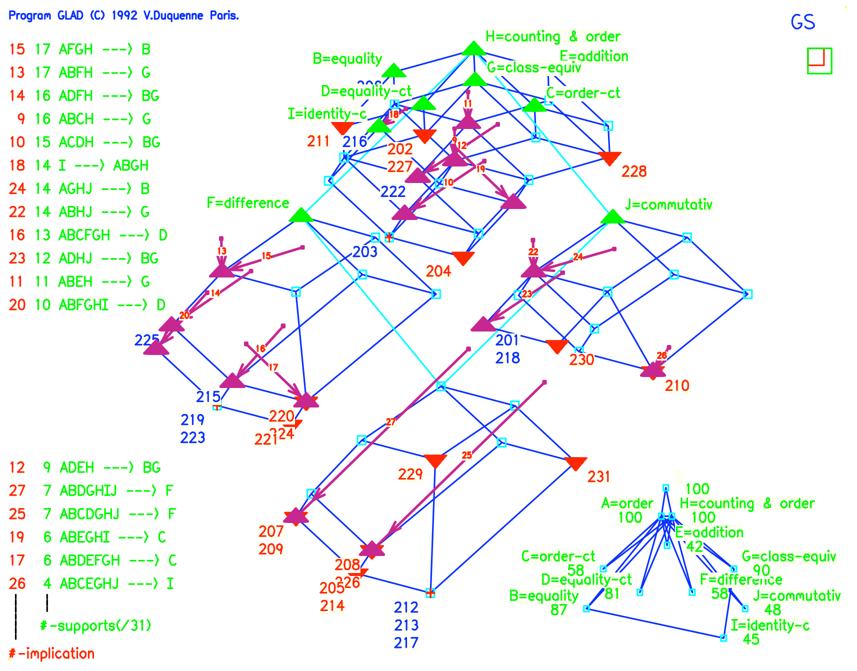
Fig. 2. Unglued decomposition in the direct product of 2x3 intervals. Each interval groups together the attributes of similar difficulties. Dually the children are grouped and scaled in homogeneous groups regarding their level of performance.

distinct directions, *order-ct* above *addition* on the one hand, and (*idendity-c*, *equality-ct*) on the other, which are themselves refined by *commutativ.* then *difference*. Notice that in that 2x3 direct product of intervals, *commutativ.* is below *order-ct*, while *difference* is below all intervals, hence with maximum difficulty level for that group. However this doesn't mean that *difference* is made comparable with *addition* in the lattice order relation: it is below in the 2x3 lattice of intervals i.e. for the "macro" difficulty levels that are made similar & scaled by the similarity (tolerance) relation.

The 2x3 interval product structures the (pre)order of children in terms of similar level of performance. The maximum element of the bottom interval is the lower cover of the element labeled 121 (close to the top) of which the intension is *ABH*, and extension comprises 23 out of the 31 children. This means that the five other intervals structure the nine remaining children, some of which being in difficulties: for instance child 121 (at the top) may require special attention and support, and the same is true for kids 115 / 103 (top right), or 122 / 105 / 116 / 177 (top left), but with different programs. An interesting outcome of this analysis could be to make proposals to constitute homogeneous groups of children of "similar level of performance".

A teacher aware of this information in real time could consequently define strategies for planning games and exercises for training the kids regarding their specific needs. Besides personalized cares as before, the teacher may want to train bigger groups of children, if possible made homogeneous. For example, the fifteen children that belong to the interval containing child 116 and *commutativ.* (bottom left, between child 107 and *equality* & *counting*) share the property of "not mastering *class-equiv.* and *addition*": all of them should benefit of special games / training specifically oriented to master these two attributes. Hence, looking within intervals reveals the specificity of performances for children of similar level, while comparing intervals can focus on what they miss or require to collectively make further progress.

For the second group ("GS", age 5.5 years) the global structure is quite different. *Counting* and *order* are now mastered by all the children, hence become superfluous regarding the lattice structure, and create eight obvious implications (expelled from the basis, see Fig 3, top left). There is only one simple implication left *identity-c* $\rightarrow$ *equality* & *class-equiv.*, a rather poor order structure between attributes, although the lattice is quite small (61 out of the potential 2^8=256 elements). Five attributes are mastered by more than 87% of the children, *commutativ.* and *difference* are no more under-represented, with now 48% and 58%, respectively. This contributes to the fact that the lattice has many elements below them: they associate in similar ways with other attributes (as opposed to what is occurring for "MS"-group), along nearly isomorphic intervals (ideals). Out of the twelve implications whose extensions are at least ten, after having removed superfluous A, H from their premises, nine have two attributes (ex: FG $\rightarrow$ B). As compared with the younger "MS"-group, it could be said that the general structure for the group of "GS"-children explodes in the direction of pairwise independence between attributes, almost without order relationships between them, but global independence and combinatorial explosion are tamed and reduced by a series of rather simple premise implications going in all directions. This explosion will perhaps not surprise teachers and parents who know that *grande-section*, which is the last year of *école maternelle* before the more academic universe of *école primaire*, is a special turn in the curriculum of children, a year of all discoveries…

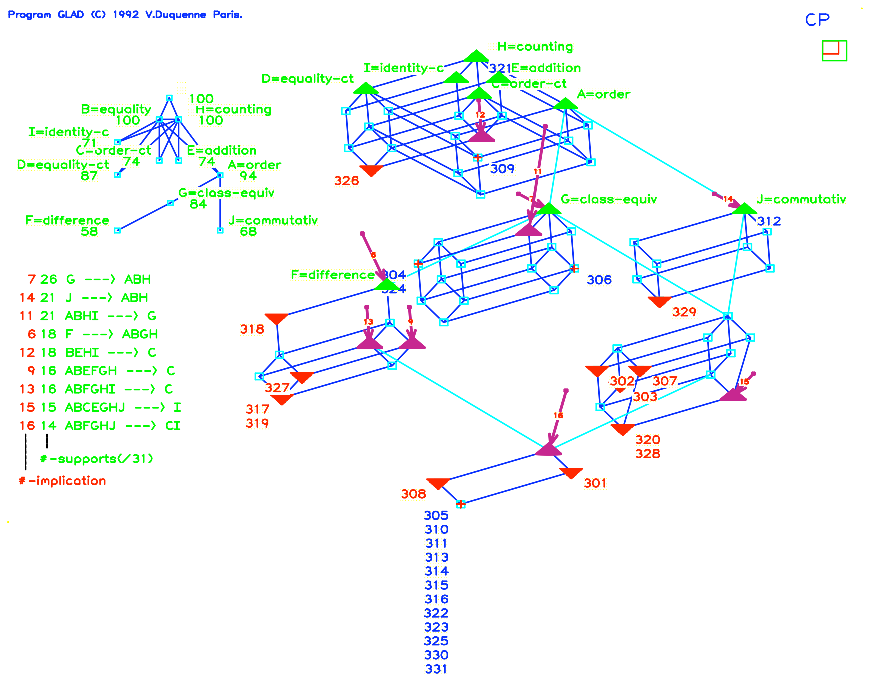


```
Program GLAD (C) 1992 V.Duquenne Paris.

15  17  AFGH ---> B
13  17  ABFH ---> G
14  16  ADFH ---> BG
 9  16  ABCH ---> G
10  15  ACDH ---> BG
18  14  I ---> ABGH
24  14  AGHJ ---> B
22  14  ABHJ ---> G
16  13  ABCFGH ---> D
23  12  ADHJ ---> BG
11  11  ABEH ---> G
20  10  ABFGHI ---> D

12   9  ADEH ---> BG
27   7  ABDGHIJ ---> F
25   7  ABCDGHJ ---> F
19   6  ABEGHI ---> C
17   6  ABDEFGH ---> C
26   4  ABCEGHJ ---> I
 |   |
 |   # -supports(/31)
 |
# -implication
```

Fig. 3. Nested line diagram of the lattice (group 2: “GS”), with canonical basis of implications (18 out of 26, *counting* & *order* being confused with the top generate 8 obvious implications) and supports. Here, only $(I \rightarrow ABGH)$ has a simple premise.

As for the third group (“CP”, age 6.5 years), 12 kids (out of 31) master all the attributes (to be compared with only one and three for the younger “MS” and “GS” groups, respectively), and most of the others are close -even very close- to the bottom. This is reassuring: a majority of these children now master almost all the 10 attributes. Moreover, five attributes receive more than 80%, while the others get between 58%-74%. After removing the three superfluous attributes (confused to the top with 100%), it remains only nine implications in the canonical basis, with only *difference* → *class-equiv*. → *order* and *commutativ*. → *order* which scale and order the attributes (see Fig. 4). These three simple implications control the lattice unglued decomposition downwards (see Fig. 5), in a chain of three totally ordered intervals.

At the top, there are only two children (321,326) who do not master *order* –and a fortiori the attributes implying *order*- and are somehow in difficulty: they should be helped specifically. The middle interval (between children 320,328 up to *order*) contains 10 children. They should be trained about *difference* to try collectively to master it and move down to the bottom interval. Downwards there are 18 children who all master *difference*, which is still the more difficult attribute for this older group. In this bottom interval, a small group of five kids (306,317,318,319,327) may have an interest in working first *commutativ*. As compared with “GS”, this group is far from displaying independence, and the analysis shows that three subgroups could be profitably distinguished to be first trained on *order*, or *commutativ*. and *difference*.



```
 7 26  G ---> ABH
14 21  J ---> ABH
11 21  ABHI ---> G
 6 18  F ---> ABGH
12 18  BEHI ---> C
 9 16  ABEFGH ---> C
13 16  ABFGHI ---> C
15 15  ABCEGHJ ---> I
16 14  ABFGHJ ---> CI
 |  |
 |  #-supports(/31)
#-implication
```

Fig. 4. Attribute preorder and the canonical basis of implications (9/16, *equality* & *counting* being confounded). There are only three simple implications.

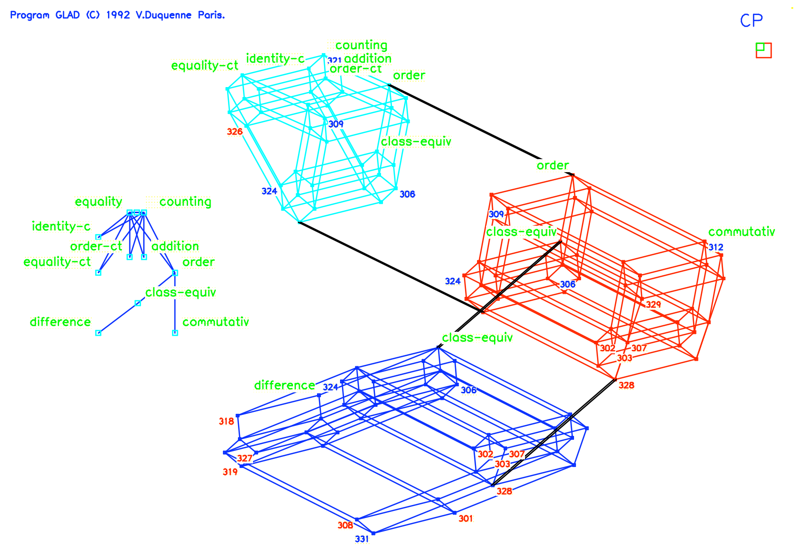


Fig. 5. Unglued decomposition in three intervals down-generated by the implications: *class-equiv*, *commutativ*. → *order* and *difference* → *class-equiv*, respectively.

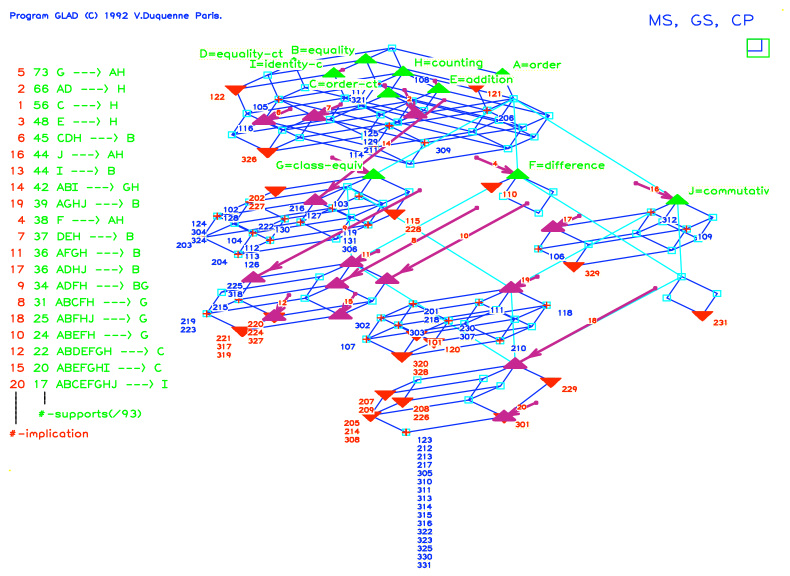
Inter-group analysis and conclusion

The three groups of children have been put together in a $C_{93}xA_{10}$ table, for which the lattice L_{123} is represented by a nested line diagram in Fig. 6. As for the previous "CP" group, the lattice appears to be linearly unglued in three intervals, where the middle one –between child 204 to *counting* see Fig 7- comprises 35 children, and can be seen as down generated by *order-ct* (60%) and *addition* (52%). The top one is up generated by children 121,122 who do not master *counting*, while the bottom interval can be seen as down generated by the two challenging attributes *commutativ.* (47%) and *difference* (41%). This structure scales the children in three levels of performance: to master *counting* or not, and the same question with *commutativ.* and *difference.*

Now, it is interesting to scan through all children' labels to notice first that older children have a tendency to be located down in the lattice: they know more, fortunately... But to study this distribution of ages spread along the lattice may also give teachers the idea to organize small groups gathering together children of different forms, because they would have the same or homogeneous performances. Hence, the interval between child 204-*class-equiv.* comprises children of the three forms who all master *class-equiv.* but not *commutativ.* nor *difference*, so that they could benefit from activities to get them. This may help in organizing some "groupes de niveaux" (level groups) in a same school, in the direction of more open education.

Coming back to intensional questions, one may raise the following one: because of the institutional gap between *école maternelle* and *école primaire*, but also due to the kind of explosion that was stated for the "GS" group, could it be possible to summarize in a compact way the shift between groups 1+2 ("MS"+"GS") and group 3 ("CP")? A direct answer is available with the notion of *relative canonical basis of implications* (see [Do&Al01] and GLAD [83-96]) expressing how the lattice L_{123} –mixing together the three groups- is projected onto the lattice L_3 (which is a sub-semi-lattice of L_{123}), and that will be denoted by $B_{3/123}$. We have used this notion for years, specially in genetics. The relative basis $B_{3/123}$ is listed in Fig. 8 where each implication is weighed by its "123"-premise's extension and "3"-intension's extension, so that their difference shows how many "12"-children will be pushed down onto the 3-intension. The first two implications come partly from the fact, that B and H are superfluous with 100% among "3"-children, and assess (see Fig. 8) that respectively 55 and 60 "12"-children are "pushed" to BH from B and H. Similarly, 57 and 43 "12"-children are "pushed" to ABH and DBH. Many of the remaining implications contain ABH in their premise, so that BH can be erased to get simple premise, since $A \rightarrow BH$. Hence the fifth implication may be simplified to A:*order* & F:*difference* → G:*class-equiv.* which concerns 19 children. Interestingly the five remaining implications in the basis have only C:*order-ct* or I:*indentity-c* –or their conjunction- as a conclusion. Thus, the gap between "MS"+"GS" and "CP" can be characterized with a few implications expressing either that *equality* and *counting* are fully mastered, or how *class-equiv.*, *order-ct* and *identity-c* are better mastered in "CP".

Before concluding, it should be added that the lattice L_{123} has a structural property which is interesting, even if it could not be seen easily but was discovered through a program: out of its 124 intensions, 114 have a unique *minimal generator*. In words, for any such intension $B^{\downarrow\uparrow} \subseteq A$ there is exactly one subset $Bo \subseteq B^{\downarrow\uparrow}$ such that $Bo^{\downarrow\uparrow} = B^{\downarrow\uparrow}$



```
Program GLAD (C) 1992 V.Duquenne Paris.

 5  73  G ---> AH
 2  66  AD ---> H
 1  56  C ---> H
 3  48  E ---> H
 6  45  CDH ---> B
16  44  J ---> AH
13  44  I ---> B
14  42  ABI ---> GH
19  39  AGHJ ---> B
 4  38  F ---> AH
 7  37  DEH ---> B
11  36  AFGH ---> B
17  36  ADHJ ---> B
 9  34  ADFH ---> BG
 8  31  ABCFH ---> G
18  25  ABFHJ ---> G
10  24  ABEFH ---> G
12  22  ABDEFGH ---> C
15  20  ABEFGHI ---> C
20  17  ABCEFGHJ ---> I
 |   |
 |   #-supports(/93)
#-implication
```

Fig. 6. Nested line diagram of the lattice (groups 1,2,3: 93 kids) with canonical basis of implications and their supports. A child's first digit indicates his / her group (1-3).

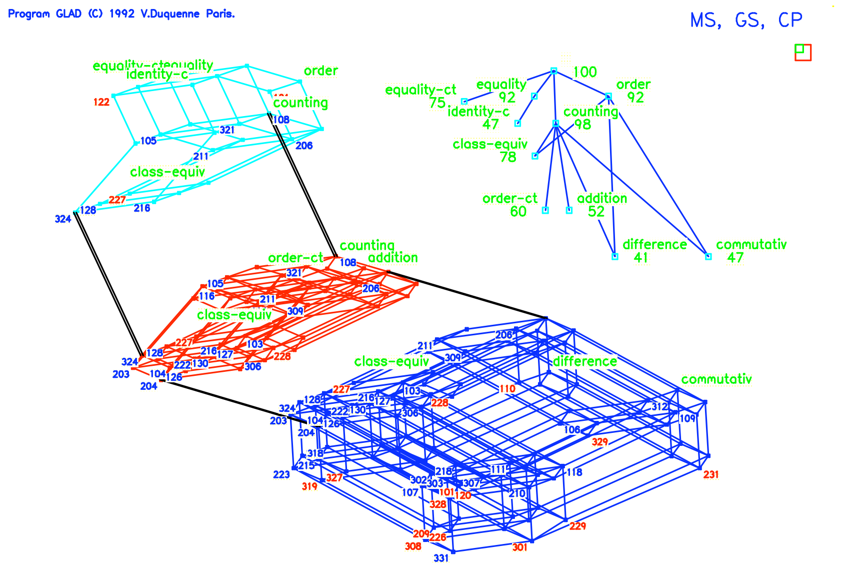


Fig. 7. Attribute order and lattice decomposition into three intervals, generated by *order-ct*, *addition* → *counting* and *difference*, *commutativ*. → *counting* & *order*.

and which is minimal for this property. Consequently, the lattice is “nearly” *join-distributive* (see GW99 p. 228), which means that collectively, the children’ performances behave “nearly” like if it was structured on an abstract *convexity space*. A lot of literature on convexity lattices has been written. They have many different characterizations (see for instance [M85]). In particular in such a convexity lattice, for any $Bo^{\downarrow\uparrow}$ and its unique minimal generator Bo, ($Bo^{\downarrow\uparrow}\backslash\{b\}$) is an intension for all “*extreme points*” b of Bo, so that the interval up-generated by the upper covers of $Bo^{\downarrow\uparrow}$ is Boolean. This expresses a property of *local independence* of extreme points. In other words the implication $Bo \rightarrow Bo^{\downarrow\uparrow}$ indicates that the set Bo of “extreme points of $Bo^{\downarrow\uparrow}$” implies what is collected “inside” their *hull* for the convexity structure.

The convexity structure provides a model for these three group behavior, but that can be also formulated as a model for each individual: a child $c \in C$ belongs to such or such a group’s extension -say $B^{\downarrow}$ -if and only if the children “c” masters a specific and unique set of keys (“extreme” attributes Bo). This provides a fine description of the minimal prerequisites which are sufficient although necessary for mastering these mathematical concepts and finding one’s path in the natural number construction.

```
 Relative basis B_3/123
86 31 55 B ---> H
91 31 60 H ---> B
86 29 57 A ---> BH
70 27 43 D ---> BH
37 18 19 ABFH ---> G
27 18  9 BEHI ---> C
24 16  8 ABEFGH ---> C
27 16 11 ABFGHI ---> C
21 15  6 ABCEGHJ ---> I
25 14 11 ABFGHJ ---> CI
 |  #3 |
 |    (#123-#3)
 #123
```

Fig. 8. The relative canonical basis of implication $B_{3/123}$ summarizes how the elements of L_{123} are "pushed down" onto its (sub-semi)lattice L_3 in a minimal way. The difference of extents (#123-#3) indicates the number of such children (groups 1,2).

For concluding, this note re-analyzes data on the construction of natural numbers in children of 4.5-6.5 years old, with a descriptive method based on orders and lattices. The three groups behave quite differently in terms of attribute orders. While the “GS” order is almost an antichain of pairwise incomparable attributes, the two others -“MS” and “CP”- have a rich structures which are emphasized as they generate unglued decompositions of the lattices. Surprisingly the same properties apply to the lattice L_{123} that mixes together the three groups. This provides a scaling of the attribute orders in subsets of similar difficulties. Dually this structures the children in terms of homogeneous sub-groups, which would be most interesting for training the kids on specific target attributes. In the same manner a close examination of L_{123} can give rise to proposals for defining small groups mixing together children of different forms in the direction of a more open school. The shift between “MS+GS” to “CP” has been characterized through a relative basis of implications, and L_{123} behaves like convexity lattices that provides minimal keys as a model for describing child performance.

Acknowledgements

First of all many thanks are due to Dr. Camilo Charron, for sharing his data and for interesting discussions when he was preparing his thesis, long ago. At this occasion, preliminary lattice analysis were done, although GLAD was not that developed in these days... A first draft of this note was discussed at "Le treillis Rochelais", La Rochelle, May 2007, for which the organizers and attendants are kindly thanked. The discussion gave the idea to write a first note, as "neutral" as possible, and try to interpret further the finding afterwards, in a potential sequel. Thanks are also due to the referees for their remarks and comments which helped in clarifying the paper.

References

[Bi67] Birkhoff G., *Lattice Theory*, Amer. Math. Soc. Colloq. Publ. Vol.25, third edition, Amer. Math. Soc., New York, N. Y. 1967 (first ed. 1940).

[Bi70] G. Birkhoff, What can lattices do for you?, in *Trends in Lattice Theory* (J. C. Abbott ed.), Van Nostrand, New York, 1970.

[C&Al01] Chabert C., Cherfouh A., Delabar J.-M. and Duquenne V., Assessing implications between genotypic / phenotypic variables through lattice analysis, *Behavior Genetics 31* (2001) 125-139.

[Cha98] Charron C., Ruptures et continuités dans la construction des nombres, Thesis (1998), Université Paris V-René Descartes, Paris.

[Do&Al01] Doyen A.-L., Duquenne V., Nuques S. and Carlier M., What can be learned from a lattice analysis of a laterality questionnaire?, *Behavior Genetics 31* (2001) 193-207.

[D84-87] Duquenne V., Contextual implications between attributes and some representation properties for finite lattices. in *Beitrage zur Begriffsanalyse*, (B. Ganter, R. Wille and K.E. Wolf eds), 1987, Mannheim: 213-239.

[D83-96] Duquenne V., GLAD (General Lattice Analysis & Design): a FORTRAN program for a glad user, *ORDAL 96*: *Order and decision-making* (I. Rival ed.), Ottawa, www.csi.uottawa.ca.

[D86] Duquenne V., What can lattices do for experimental designs?, *Mathematical Social Sciences 11* (1986) 243-281.

[D99] Duquenne V., Latticial structures in Data Analysis, *ORDAL 96*: *Order and decision-making* (I. Rival ed.), Ottawa, www.csi.uottawa.ca, published in *Theoretical Computer Science 217* (1999) 407-436.

[D&Al01] Duquenne V., Chabert C., Cherfouh A, Delabar J.-M., Doyen A.-L and Pickering D., Structuration of phenotypes / genotypes through Galois lattices and implications, in *Proc. of ICCS2001-CLKDD'01* (Stanford 07/2001, E. Mephu Nguifo et *al.* eds) 21-32, and *Applied Artificial Intelligence 17* (2003) 243-256.

[G84-87] Ganter B., Algorithmen zur Formalen Begriffsanalyse in *Beitrage zur Begriffsanalyse*, (B. Ganter ed.), 1987, Mannheim: 213-239 (preprint TH Darmstadt, 1984).

[GW99] Ganter B. and Wille R.. *Formal Concept Analysis, Mathematical Foundations.* Springer Verlag, Berlin, 1999.

[GD84-86] Guigues J.L. and Duquenne.V., Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathematiques & Sciences Humaines 95* (1986) 5-18 (preprint Groupe Mathématiques et Psychologie, Université Paris V-René Descartes, 1984).

[M85] Monjardet B., A use of frequently rediscovering a concept, *Order 1* (1985) 415-417.

[O44] Ore O., Galois connexions, Trans. Amer. Math. Soc. *55* (1944) 493-513.]

[Wi82] Wille R., Restructuring lattice theory: an approach based on hierarchies of concepts. In *Ordered sets* (I. Rival ed), Reidel, Dordrecht-Boston (1982) 445-470.

Using Concept Lattices for Visual Navigation Assistance in Large Databases: Application to a Patent Database

Jean Villerd[1], Sylvie Ranwez[1], Michel Crampes[1], David Carteret[2].

[1] Centre de Recherche LGI2P de l'Ecole des Mines d'Alès, Site EERIE, Parc scientifique G. Besse, 30 035 Nîmes, France
{Jean.Villerd, Sylvie.Ranwez, Michel.Crampes}@ema.fr

[2] I-Nova, 11, avenue Albert Einstein, 69100 Villeurbanne – Lyon, France
david.carteret@i-nova.fr

Abstract. The increasing size of indexed document sets that are digitally available emphasizes the crucial need for more suitable representation tools than the traditional textual list of results. Many efforts have been made to develop graphical tools able to provide both overall and local views of a collection when focusing on a particular subset of documents. However an overcrowded visual representation may not be really useful to users if they are not guided in their navigation process. Our goal is to combine the classification features of FCA and existing visualization techniques to suggest navigation paths in a visual representation through meaningful and progressive focuses. The test collection used for our study is an indexed patent base provided by our industrial partner.

Keywords: FCA, Visual navigation, Semantic navigation.

1 Introduction

The size of digitally available indexed document sets increases every day. However, associated exploring tools are often based on the same traditional model: users send their query and are then answered back with huge lists of results. There is a crucial need for more suitable representation tools where the semantics of the documents are better exploited and may be used as a guideline during navigation through the database. Formal concept analysis (FCA) helps to form conceptual structures from data. Such structures may be used to visualize inherent properties in data sets and to dynamically explore a collection of documents. Indeed the associated mathematical formalization is useful not only to organize the database but also to infer some of the reasoning during the information retrieval process. This paper presents a method that combines FCA and Information visualization techniques to assist visual navigation in large collections. This research has been done in response to an industrial demand: one of our partners is facing the problem of visualizing and browsing a large collection of indexed patents.

The following section will present the context of our research. The state of the art is presented in section 3, particularly concerning information retrieval using FCA and visualization techniques to explore large databases. Section 4 develops our method that combines FCA and visualization tools to assist users for browsing a large collection. Section 5 deals with our current work on a real database provided by our industrial partner. The last section concludes with some of the limits to and perspectives of our approach.

2 Context and Problem Setting

Searching for technical solutions to improve innovation within big companies, I-Nova[1], our industrial partner, develops collaborative platforms to internally share some parts of the company's knowledge. It may be sets of *ideas*, patents, laws, etc. The efficiency of this sharing relies on the participation of every type of employee. Therefore the interface must be intuitive enough to favor the participation of people not familiar with such environments. The major problem consists in visualizing and browsing a large collection of indexed documents. The current platform makes use of a classical search engine which lists retrieved indexed documents (patents, here) corresponding to a certain set of keywords. Two main problems arise: human operators cannot have an overall view of the whole collection and they cannot evaluate changes in the result sets when adding or removing a keyword, because a new list is displayed for each new query. These drawbacks are particularly semantically noisy for the browsing process. We may note here that this problem is the same as the one encountered by people using web search engines like Google.

Much research has been done to graphically represent indexed document sets in general [30], of which several aim to represent patent databases. In MultiSOM [16], the keywords are divided into subsets corresponding to different aspects of the indexation (costs, techniques, etc.) and a self-organized map is computed one by one for each subset, presenting different points of view on the database. When focusing on a specific item of the collection on one particular map, the user can switch to another map presenting the position of the item in the collection with respect to another aspect. This solution solves the problem of providing an overall view of the collection. However users have lost the ability of selecting a subset of patents through a set of keywords and they have no information about why one patent is close to another in the overall view.

Because on the one hand browsing an overall view may be unsuitable for focusing on a particular keyword and on the other hand displaying local results without any overall information causes users to get lost, we present in this paper a method that assists users' navigation from overall to local views. The idea is to "sum up" the collection by coherent local views corresponding to subsets of patents/keywords. These views are ordered as a lattice, defining possible navigation paths that will be suggested to users.

The views' lattice is actually a concept lattice [12]. Using FCA and concept lattice for information retrieval is not new [10] but in these approaches, contents of nodes,

[1] http://www.i-nova.fr/

i.e. local view, are still displayed as a list of results, retaining all the above mentioned problems. Studies on visualization exploration process have been done but mainly in the medical or scientific imagery domain, focusing more on optical transitions between visualizations [14] than on semantic aspects of the information that is displayed. Before going further towards the solution that we propose, let us give some basic FCA definitions.

3 Formal Concept Analysis Background

In this section we briefly recall FCA basic definitions from [2][12].

A *formal context* is a triple (G, M, I) where G is a set of objects, M a set of attributes and I is a binary relation between the objects and the attributes, i.e. $I \subseteq G \times M$.

For a set $O \subseteq 2^G$ of objects and a set $A \subseteq 2^M$ of attributes, we define the set of attributes common to the objects in A, by

$$f : 2^G \rightarrow 2^M \qquad f(X) = \{a \in A \mid \forall\, o \in X, (o,a) \in I\}$$

the set of objects which have all attributes in O, by:

$$g : 2^M \rightarrow 2^G \qquad g(Y) = \{o \in O \mid \forall\, a \in Y, (o,a) \in I\}$$

The pair (f, g) is a Galois connection between $(2^G, \subseteq)$ and $(2^M, \subseteq)$.

A *formal concept* of the context (G, M, I) is pair (O,A) with $O \subseteq 2^G$, $A \subseteq 2^M$ and $A = f(O)$ and $O = g(A)$. A is called the *intent* and O the *extent* of the concept (A,O).

Let L be the set of concepts of (G, M, I) and let $\leq_L$ be a partial order defined as follows: $(A_1,O_1) \in L$, $(A_2,O_2) \in L$, $(A_1,O_1) \leq_L (A_2,O_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow O_2 \subseteq O_1$
The pair $(L, \leq_L)$ is called the *Galois lattice* or *Concept lattice* of (G, M, I).

The *simplified extent* of a concept *(O,A)* is the set of objects which belong to O and do not belong to any lower concept. In other words, the simplified extent denotes objects that do not have any other attributes than those in A.

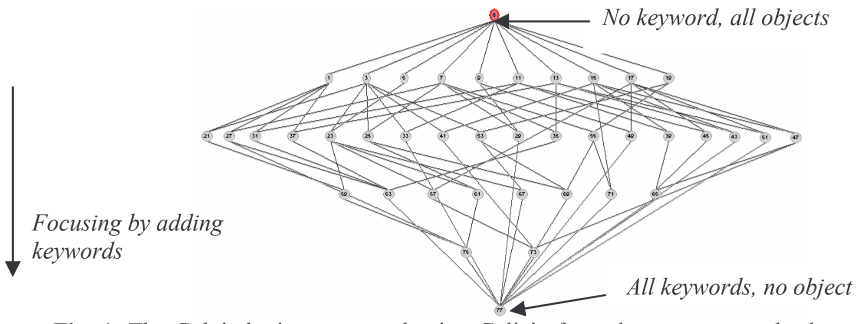


Fig. 1. The Galois lattice computed using Galicia from the patents test database (329 objects, 10 attributes).

In the following we denote indifferently objects as documents or patents and attributes as terms or keywords. The Galois lattice computed from the patent/keyword matrix of our test database is shown in Fig. 1.

4 State of the Art

Searching for a solution to assist the navigation through a large database, we focus particularly on two aspects in the following state of the art: applications that use FCA techniques for information retrieval and visualization techniques that may be used to graphically parse large sets of data.

4.1 Formal Concept Analysis for Information Retrieval

The powerful classification skills of Formal Concept Analysis have found many applications for information retrieval. Some of them have been listed in [23]. Since the early works of [13] on an information retrieval system based on document/term lattices, a lot of research leading to significant results has been done. In [7], Carpineto and Romano argue that, in addition to their classification behaviors for information retrieval tasks, concept lattices can also support an integration of querying and browsing by allowing users to navigate into search results. Nowadays several industrial FCA-based applications like Credo [7] or Mail-Sleuth [11] are available. Mail-Sleuth is an e-mail management system providing classification and query tools based on FCA. This tool allows users to navigate into data and intervene in the term classification by displaying concept lattices. Upstream research has studied the understandability of a lattice representation by novice users [11]. Image-Sleuth [10] proposes an interactive FCA-based image retrieval system in which subjacent lattices are hidden. Users do not interact with an explicit representation of a lattice. They navigate from one concept to another by adding or removing terms suggested by the system. This ensures a progressive navigation into the lattice.

4.2 Visualization Techniques to Browse Large Databases

Graphical solutions for visualizing a mass of abstract information have been studied for several decades, leading to the emergence of the information visualization domain [5]. Even with the use of a visual representation, the navigation into a large collection may not be obvious. Schneiderman has defined a visualization information paradigm called “focus + context” which recommends to first provide an overall view, then to let the user identify an area of interest (focus) and finally to display locally contextual information (context) [25]. Starting from an overall view helps users to maintain a unique mental map but they still have to achieve the focus task on their own.

In the particular case of document visualization, [5] defines two categories of solutions: on the one hand visualizations of the inner structure of a document (WebBook [6]) and on the other hand visualizations of a document collection, e.g. DocCube [20]. The work presented in this paper belongs to this second category which can also be divided into two parts depending on how the structure of the collection is managed. Some tools show this structure by representing clusters of documents (e.g. Grokker[2]) using tiling based visualization techniques such as TreeMaps [15]. Some other tools

[2] http://www.grokker.com

do not show the collection's structure and represent the collection by a set of points into two or three dimensions dispatched according to a semantic distance usually based on indexed vectors (e.g. DocCube).

The choice between these two strategies depends on the users' needs. Showing the structure allows a more progressive navigation through clusters but reduces the probability of visual insight which the observation of similarity distances may offer. We have tried to benefit from both solutions by using the first one for the overall view and the second one for local views (see section 5.3).

4.3 Visualization Paths

The visualization exploration process has been studied for several decades. Upson et al. [31] and Card et al. [5] describe the visualization process as an analysis cycle, starting from raw data transformation, mapping onto the visual primitive, and view rendering provided to the user who performs a feed-back that restarts the cycle. The central idea of their models is that a description of the visualization process has to focus on this interaction with the user who changes the visualization parameters. The user influences all steps of the process by adjusting these parameters: from data filtering and transformation (by specifying a data threshold value), mapping onto visual primitives (by specifying another color or shape) and view rendering (by changing the orientation of the view). Upson and Card's models provide a characterization of visualization exploration as a whole global process but their granularity is not deep enough to detail and define a particular user's exploration path. Following their work two approaches have been taken: visualization space paths and derivation models.

Visualization space paths [29] consider user's visualization exploration as a navigation path in a multidimensional parameter space, a visualization result being a single point in the parameter space. Based on this model, novel visualization user interfaces such as Design Galleries [19] display an overview of the entire parameter space by random sampling. The main interest of this model is that it provides an explicit relation between visualization results and visualization parameters. However, particular relations between results are not explicit. Derivation models have been introduced to overcome this problem.

Derivation models describe how new visualization results are created from previous ones. Originally developed to provide a visual assistance to users of scientific problem solving environments, the GASPARC system [3] builds a history tree of visualization results in order to store solution parameters and related results. In Lee's work [17][18], a graph structure is used to model the visualization process for databases. Vertices stand for visualization states and edges are based upon similarities between structural attributes of the states. Jankun-Kelly et al. [14] extend these models to a more general and extensible one.

Due to the historic development of visualization applications, the majority of these works concentrate on scientific visualization (problem solving, medical imagery), rather than information visualization. Unfortunately, the semantics of the visualized data, which is what we are interested in, are then not taken into account in the cycle of the visualization exploration process.

4.4 Knowledge Maps and Multidimensional Scaling

The tests and validations of our approach are done through an agent oriented software environment that we developed. Molage (for Molecular Agents) allows visual manipulation of entities using several visual functionalities (zoom, fisheye, semantic lenses, filtering, etc.). Each entity may be typed and described by several attributes (descriptors). The collection of those entities may represent a multimedia database, e.g. music titles described by moods, textual documents characterized by a key-words' vector, pictures that are classified according to their subject, etc.

Considering the set of descriptors, each entity is characterized in a *m* dimensions' space. The collection is projected onto a 2 dimensional plane using a MultiDimensional Scaling projection (MDS – [22]). The method consists in minimizing a 'stress' function between *n* points (originally described in an *m*-dimensional space) after those points have been projected onto a space with fewer than *m* dimensions. The minimization is performed through the compression or extension of the (Euclidian) distances between the points in the smaller dimension [4]. This type of MDS approach is known as Force Directed Placement or Spring Embedding Algorithm.

We detailed the Molage environment and its use for navigating through a massive music collection in [8]. In this particular application, the musical landscape is used to semi-automatically index new music records just by "drag and drop" of a new entity on the map. It is also used to automatically build musical playlists. However some drawbacks were still present, in particular the lack of assistance for the visual navigation through the map. That is why we proposed to give some semantics to the map and to assist the building of this map in [9]. In the following, we broaden this visual assistance by proposing a method for browsing large databases.

5 A Visual Database Browsing Method

Within large databases, it is often possible to distinguish several layers that may be considered separately. Our idea is to keep the human operator's mental map as stable as possible and to give him or her some hints when he or she switches from one view to another one.

5.1 Problem Decomposition

Our approach aims to provide adapted visualizations for different abstraction layers. Information visualization techniques aim to assist the user in a visual inference task. It is very hard to reach a compromise between a strongly analytic strategy displaying results of classification methods on data and a visualization representing more faithfully all raw data. The first approach may be clearer for users but may introduce a bias in their interpretation and preclude insight by hiding unexpected information. The second approach faces the dimensionality problem of visualizing too much information at the same time. Our work proposes to satisfy both approaches by finding a compromise.

We assume that Schneiderman's overview, i.e. the overall view corresponding to the higher abstraction layer, has to emphasize the structure of the document collection because that is where the source of insight may reside at this level. This overview should be used as a kind of GPS for the further navigation into different local views. For the lower abstraction level, i.e. Schneiderman's context views, the aim is to display concrete relations between local data using the most intuitive visualization techniques. In the following we will describe the overall view and then local views but note that we propose a user interface that will show the overall view and one particular local view at the same time (see Fig. 3).

5.2 Lattice-based Overview

The concept lattice computed from the document/term (patent/keyword, for us) matrix will be used both as a support for navigation across the collection (as used in Image-Sleuth), and as a visual overview emphasizing its structure.

Rather than using traditional lattice representation such as Hasse diagrams (see Fig. 1) and graph drawing techniques [28], lattice nodes are displayed using force direct placement techniques (see section 4.4) according to a semantic distance [24].

All edges may not be represented as in Fig. 2 (left) to avoid visual overloading. We use a visual device called "topological lenses" [9] to show edges of the selected node and to emphasize nodes reached by these edges. In Fig. 2 (right), the selected node's intent is {plasma display, display device} and its simplified extent's cardinality is 1 (this cardinality indicates how many documents belong to this concept and not to any lower). Relied concepts and their intents are emphasized. Moving the cursor over a concept shows its intent. This visualization allows users to browse nodes through lattice order while evaluating the semantic distance covered at each step.

This visualization allows users to browse nodes through lattice order while evaluating the semantic distance covered at each step thanks to the similarity distance.

Several strategies could be applied in order to reduce the number of nodes to avoid visual overloading. Iceberg lattices [27] and alpha lattices [21] consist of smaller concept lattices retaining significant concepts that respect a threshold criterion based on the extent cardinality. Considering the concept lattice as an overview of the collection, these lattice reduction techniques based on the frequency of the concepts are compatible with our approach. Another lattice reduction strategy would consist in decomposing the overall lattice into smaller sub-lattices, as performed in Image-Sleuth. Partition of term space would be achieved by a domain expert or by identification of clusters of concepts with respect to the semantic distance.

Since an overview supporting navigation is already provided to users, we will present in the following how a local view associated with a lattice concept is visualized and then how users interact with local and overall views.

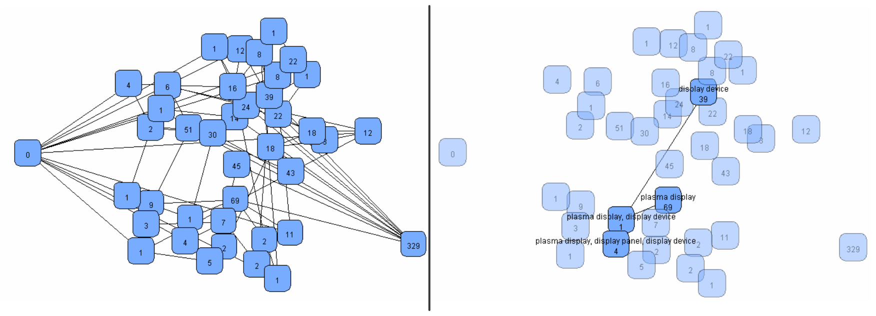

display device
plasma display
plasma display, display device
plasma display, display panel, display device

Fig. 2. The Overall view is a MDS projection of the lattice concepts with respect to a semantic distance. All edges shown (*left*). A particular concept (plasma display, display device) and his neighbors emphasized (*right*).

As presented in section 5.1 local views should represent relations and similarity between raw data. An alternative possibility concerns the use of a similarity distance between documents.

A first approach consists in computing a distance between documents once and for all. Much research has been done concerning semantic distance computation between textual documents which are outside the scope of the present paper [24]. We assume that a distance is available between documents, e.g. the semantic distance proposed in [24].

This distance may be independent from the information used as attributes in the formal context. Indeed it can be interesting to compare two indexation layers. For instance, in the case of multimedia documents, textual tags such as musical moods could be used to structure the collection and build the concept lattice whereas physical descriptors from signal processing analysis such as Mel Frequency Cepstral Coefficient [1] could be used to compute a similarity measure between documents, assuming that users' research pattern consists in using high abstraction layer features to achieve Schneiderman's focus task and then using low abstraction layer, closer to raw data, to observe local organization of information.

Using the same distance to represent all local views ensures users' mental map preservation. Actually, moving from one lattice concept to another consists in making some documents visible or not, so remaining documents' positions should not change even if the heuristic nature of Force directed placement approach may affect stability. Anyway Molage is able to fix positions for a subset of objects (here remaining documents), letting others (here newly retrieved documents) find a position with respect to semantic distance.

Another approach consists in computing a particular distance for each lattice concept, i.e. for each set of formal attributes. Assuming that each term is associated to a numeric function, each document has a numeric vector on which a MDS projection can be performed, taking into account rows corresponding to terms present in the lattice concept's intent. Moving from one lattice concept to another also consists in making some documents visible or not but positions of all documents change because a new distance is computed by adding or removing vector rows in the MDS calculation. Even if this approach provides a more precise distance adapted to local concept intent, its major drawback is that users' mental maps can become less accurate. However, Molage can dynamically perform such a selective MDS projection, allowing users to observe objects' movements when updating the distance.

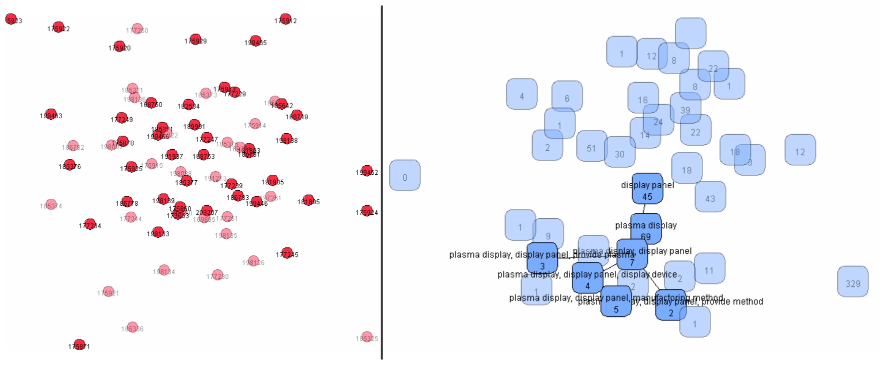

display panel
45
plasma display
69
plasma display, display panel
7
plasma display, display panel, provide plasma
3
plasma display, display panel, display device
4
plasma display, display panel, manufacturing method
5
plasma display, display panel, provide method
2
329
0

Fig. 3. The selection of a concept on the overall view (*right hand side*) displays patents contained in the concept's extent (*left hand side*) and suggests navigation paths by emphasizing new concepts on the overall view.

5.3 Interactions between Local and Overall Views

Several use cases have been identified. The most basic one consists in selecting the top node of the lattice in order to display the whole collection as a local view.
Then two possibilities occur: users can continue their navigation across the lattice by choosing one of the top nodes that has been emphasized. When users move the mouse on a particular node, documents that belong to this node are emphasized in order to preview the next visualization and to observe the distribution of these documents in the current view (see Fig. 3). When users select a new node the local view is updated to remove items that do not appear in the new node's extent. In this case, distances between documents in the different views are not recalculated and the map does not change; only the visible layer changes.

Another possibility is to select a document on the local view. The local view is then updated to visualize documents belonging to the lowest concept, i.e. the more precise, where the selected document appears. This concept is emphasized on the overall view, so that users see how far they have moved from the starting node.

6 Results

Our navigation method has been tested on real data, precisely a collection of indexed patents, provided by our industrial partner with a similarity matrix between patents. In order to reduce the size of the experimental collection, we extracted 329 patents indexed by 10 terms with an average of 0.8 term describing each patent. All these patents share the term "plasma" that does not belong to the set of 10 terms. The resulting

lattice, computed by Galicia[3] using Bordat algorithm, has 40 concepts. We submitted several screenshots to our industrial partner corresponding to different steps in a particular navigation use case. He was asked to compare the usability and user-friendliness of our method to their traditional list-based interface. Concerning the overall view, he pointed out the usefulness of the edges: "*Edges between concepts and cardinality of extents give us new information, make the navigation fun and allow us to identify the coherence of the documents related to one concept"*. Concerning the local view, he appreciated the spatialization of the documents and *"the fact that one may see at first sight how many documents are displayed in a more direct and explicit manner compared to the display of a number of results above a list of results"*. Concerning the navigation method itself, he agrees that "*it maintains a kind of mental map. That was a problem pointed out by users. The database analysis consists in two steps: an exploration step to discover and identify documents' content and then a rationalization of theses discoveries. The problem is that these two steps have to be done at the same time and with a list interface, one may forget the different discovery paths used from the beginning. The visual aspect of [our] method will surely facilitate to rationalize and save pertinent groups of documents."*
He also pointed out some drawbacks and suggested improvements: *"more information about documents should be displayed on the local view"*, *"it should be interesting to fix a local view and, when selecting a new concept, to color added documents (e.g. blue) on the local view and to differently color (e.g. violet) documents in the intersection"*. This last remark is particularly important because it matches our perspectives: to go further than navigation and to provide visual tools to assist data analysis process.

7 Conclusion and Perspectives

This paper has presented a method for visual navigation in indexed document collections combining classification behaviors of FCA to produce overviews and intuitive visualization techniques when focusing on local views. First feedbacks from our industrial partner are promising. The approach seems pertinent and results really meet the users' needs. The challenge is now to broaden the scope of our research to emphasize the analysis process during the database exploration.

As mentioned in the introduction, we aim to use FCA techniques not only to organize the database but also to infer some reasoning during the visualization navigation process. Research presented in this paper only deals with assisting users in their navigation and no information is really inferred by the system itself. This is a proper problem of information visualization. Indeed, noticing Robert Spence's definitions in [26] *"to visualize is to form a mental model or mental image of something. Visualization is a human cognitive activity, not something that a computer does"*, our goal is not to produce formal results from raw data because data analysis techniques such as FCA succeed without any visualization need. We try to explore new techniques to provide bootstraps for the cognitive activity of users. One of these would be to translate paths

[3] http://www.iro.umontreal.ca/~galicia/

in local views to paths in the overall view, in other words to express sequences of raw data by sequences of lattice concepts (each object belonging to a minimal concept).

Acknowledgments. Our particular thanks go to the I-nova company that provided us with a real patent database which allowed us to apply our research to a real problem.

References

1. Aucouturier, JJ., Pachet, F.: Music Similarity Measures: What's the Use? In Ircam, (Editor), Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR) (2002)
2. Barbut, M., Monjardet, B.: Ordre et Classification: Algèbre et Combinatoire. Hachette, Paris (1970)
3. Brodlie, K., Poon, A., Wright, H., Brankin, L., Banecki, G. and Gay, A.: GRASPARC–A problem solving environment integrating computation and visualization. In Proceedings of the 4th IEEE Conference on Visualization (Vis'93), G. M. Nielson and R. D. Bergeron, Eds. (1993) 102–109
4. Buja, A. and Swayne, D.F.: Visualization Methodology for Multidimensional Scaling. In Journal of Classification vol. 19, n° 1 (2002) 7–43
5. Card, S. K., Mackinlay, J. D. and Shneiderman, B.: Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Publishers (1999)
6. Card S. K., Hong L., Mackinlay J. D., Chi E. H.: 3Book: a scalable 3D virtual book, CHI'04 extended abstracts on Human factors in computing systems, Vienna, Austria, ACM Press, (2004)
7. Carpineto, C., Romano, G.: Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. In: Journal of Universal Computing, vol. 10, n°8 (2004) 985–1013
8. Crampes, M., Ranwez, S., Velickovski, F., Mooney, C., Mille, N.: An integrated visual approach for music indexing and dynamic playlist composition, 13th Annual Multimedia Computing and Networking (MMCN'06), San Jose, CA, US. (2006)
9. Crampes, M., Ranwez, S. Villerd, J., Velickovski, F., Mooney, C., Emery, A., Mille, N.: Concept Maps for Designing Adaptive Knowledge Maps. In Concept Maps, A Special Issue of Information Visualization, vol. 5, n° 3, Guest Editors: S.-O. Tergan, T. Keller and R. Burkhard, Palgrave - Macmillan, (2006)
10. Ducrou, J., Vormbrock, B., Eklund, P.: FCA-based Browsing and Searching of a Collection of Images. In: Proceedings of the 14th International Conference on Conceptual Structures. LNAI 4068, Springer Verlag (2006) 203–214
11. Eklund, P., Ducrou, J., & Brawn, P.: Concept Lattices for Information Visualization: Can Novices Read Line Diagrams? In: P. Eklund (Ed.), Concept Lattices: Second International Conference on Formal Concept Analysis, LNCS 2961. Berlin: Springer, (2004) 14–27
12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
13. Godin, R., Gecsei, J., & Pichet, C.: Design of Browsing Interface for Information Retrieval. In N. J. Belkin, & C. J. van Rijsbergen (Eds.), Proc.SIGIR '89, (1989) 32–39
14. Jankun-Kelly, T.J., Ma, K.-L. and Gertz, M.: A Model and Framework for Visualization Exploration, to appear in IEEE Transactions on Visualization and Computer Graphics, 13(2), (March/April 2007) 357–369
15. Johnson, B., Schneiderman, B.: Tree-Maps: a Space-Filling Approach to the Visualization of Hierarchical Information Structures. In Proceedings of the Second IEEE Conference on Visualization, IEEE Computer Society Press, Los Alamitos, CA (1991) 284–291

16. Lamirel, J., Al Shehabi, S.: MultiSOM: A Multiview Neural Model for Accurately Analyzing and Mining Complex Data. In: Proceedings of the Fourth international Conference on Coordinated & Multiple Views in Exploratory Visualization, IEEE Computer Society, Washington, DC (2006) 42–54
17. Lee, J. P.: A systems and process model for data exploration. Ph.D. dissertation, U. of Massachuesetts Lowell (1998)
18. Lee, J. P. and Grinstein, G. G.: An architecture for retaining and analyzing visual explorations of databases. In Proceedings of the 6th IEEE Conference on Visualization (Vis'95), G. M. Nielson and D. Silver, Eds. (1995) pp. 101–108
19. Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfister, H., Ruml, W., Ryall, K., Seims, J. and Shieber, S.: Design Galleries: A general approach to setting parameters for computer graphics and animation. In Proceedings of ACM SIGGRAPH 1997, ser. Computer Graphics Proceedings, Annual Conference Series (1997) 389–400
20. Mothe J., Chrisment C., Dousset B., Alaux J.: DocCube: Multi-Dimensional Visualisation and Exploration of Large Document Sets, Journal of the American Society for Information Science and Technology, JASIST, Special topic section: web retrieval and mining, vol. 7, (2003) 650–659
21. Pernelle, N., Ventos, V, Soldano, H.: ZooM: Alpha Galois Lattices for Conceptual Clustering. In: Proc. Of the Managing Specialization/Generalization Hierarchies (MASPEGHI) Workshop (2003)
22. Platt, J.C.: FastMap, MetricMap, and Landmark MDS are all Nyström Algorithms. 10th International Workshop on Artificial Intelligence and Statistics (2005) 261-268
23. Priss, U.: Formal Concept Analysis in Information Science. In: Annual Review of Information Science and Technology, vol. 40. American Society for Information Science, Washington, DC (2006) 521–543
24. Ranwez, S., Ranwez, V., Villerd, J., Crampes, M.: Ontological Distance Measures for Information Visualisation on Conceptual Maps. In Proceedings of the OTM Workshops. LNCS 4278, Springer (2006) 1050–1061
25. Schneiderman, B.: The Eyes have it: a Task by Data Type Taxonomy for Information Visualizations. In: Proceedings of the IEEE Symposium on Visual Languages. (1996) 336–343
26. Spence, R.: Information Visualization, first edition, ACM Press Books, Edinburgh Gate, Harlow, Essex, England (2001)
27. Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L.: Computing Iceberg Concept Lattices with TITANIC. In: Data and Knowledge Engineering 42, (2002) 189–222
28. Tollis I.G.: Graph drawing and information visualization, ACM Computing Surveys., vol. 28, n° 4, (1996)
29. Tory, M., Potts, S., and Möller, T.: A parallel coordinates style interface for exploratory volume visualization. IEEE Transactions on Visualization and Computer Graphics, vol. 11, n° 1, (2005) 71–80
30. Tricot, C., Roche, C., Foveau, C.E., Reguigui, S.: Cartographie Sémantique de Fonds Numériques Scientifiques et Techniques. In: Document Numérique, vol. 9, n° 2, (2006) 13–36
31. Upson, C., Faulhaber, T. A., Kamins, D., Laidlaw, D., Schlegel, D., Vroom, J. Gurwitz, R., and van Dam, A.: The application visualization system: A computational environment for scientific visualization. IEEE Computer Graphics and Applications, vol. 9, n° 4, (1989) 30–42

Evaluation of questionnaires by means of formal concept analysis[⋆]

Radim Belohlavek[1,2], Vladimír Sklenář[2], Jiří Zacpal[2], Erik Sigmund[3]

[1] Dept. Systems Science and Industrial Engineering
T. J. Watson School of Engineering and Applied Science
Binghamton University–SUNY, PO Box 6000, Binghamton, NY 13902–6000, USA
rbelohla@binghamton.edu
[2] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
{vladimir.sklenar,jiri.zacpal}@upol.cz
[3] Centre for Kinanthropology Research, Palacky University, Olomouc
tr. Miru 115, CZ-771 11 Olomouc, Czech Republic
erik.sigmund@upol.cz

Abstract. The paper presents a method of evaluation of questionnaires. The method is supported by formal concept analysis. The role of formal concept analysis consists in providing an expert with a structured view on the data collected from the questionnaires. The method resulted from experiments with IPAQ (International Physical Activity Questionnaire). The structured view on data provided by the method suggests to the expert various hypotheses which can later be tested using statistical methods. However, the structured view on data itself proved to be sufficiently informative quite often. The paper presents experiments with evaluation of IPAQ.

1 Introduction and problem setting

Questionnaires are being used in many areas of human activities. The aim is to reveal patterns of behavior and various kinds of dependencies among variables being surveyed. Descriptive statistics and statistical hypotheses testing are among the tools traditionally used for evaluation of questionnaires. A practical disadvantage of the traditional statistical approaches is the need to formulate hypotheses to be tested. Without any prior structured view on the data contained in the questionnaires, formulation of relevant hypotheses is a difficult task. Another disadvantage of traditional statistical approaches is the limitation regarding what statistics can tell about data and how statistical summaries can be understood by experts in the field of inquiry who are not experts in statistics.

This paper presents results on evaluation of questionnaires monitoring physical activities of a population and the role of formal concept analysis in this

[⋆] Supported by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, by institutional support, research plan MSM 6198959214, and by grant No. 6198959221 'Physical activity and inactivity of inhabitants of the Czech Republic in the context of behavioral changes' of the MSM.

evaluation. The paper is a continuation of previous studies regarding the IPAQ questionnaire, see [16, 17]. IPAQ is a standardized international questionnaire, see [20]. At the beginning of our study, there was a need for an alternative means of evaluation of questionnaires formulated by experts (domain experts) from the Faculty of Physical Culture of the Palacky University, Olomouc, who are involved in a world-wide project of monitoring physical activities in today's population. The experts struggled with classical statistical techniques and were looking for alternative methods of evaluation of the questionnaires. It turned out that basic methods of formal concept analysis (FCA) [11] are quite useful for the domain experts. Putting briefly, a concept lattice and its parts provide the experts with an easy-to-understand hierarchical view on the data. In terms of FCA, the basic idea is the following. The objects are the individuals (or their groups) being surveyed in the questionnaires, the attributes correspond to the variables being monitored by the questionnaires. The corresponding concept lattice or its parts reveals to the domain expert the groups in dependence on the attributes and the expert can see various dependencies between attributes, how large the groups are etc. Therefore, the concept lattice provides the expert with a first insight into the data. Such an insight is crucial. Very often, this insight is what the expert needs to see. Furthermore, based on this insight, the expert can pursue more detailed inquiries including those based on classical statistical techniques. In particular, the present study focuses on considering groups of individuals as objects. The groups are based on sharing common attributes specified by the expert. The groups can be seen as aggregates. Instead of having an attribute by an individual, we then naturally come to relative frequencies of attributes within the specified groups. This way, one comes from data tables with yes/no attributes (i.e. ordinary formal concepts) to data tables with numbers from the unit interval $[0, 1]$ interpreted as relative frequencies. We use particular fuzzy concept lattices for building concept lattices from such data tables. The concept lattices provide an expert with an aggregate hierarchical view on the data. The advantage of taking groups and the relative frequencies instead of individuals and original attributes is conciseness of the description provided by the resulting concept lattice which is what the experts asked for. The disadvantage, as with any other method which involves aggregation and summarization, is loss of information. We present our method, experimental results, as well as a brief description of the software tool we used.

2 The questionnaire

Each questionnaire consists of questions to be answered by respondents mostly by selection from possible answers. From the point of view of FCA, we can take the set of respondents as the set of objects and single questions as attributes. The questions need not be yes/no questions. Rather, some questions like those concerning age and education are many-valued. Correspondingly, a filled-in questionnaire can be represented by a many-valued context. Such context can be transformed to an ordinary formal context $\langle X, Y, I\rangle$ via conceptual scaling [11].

Typically, such a formal context contains many objects and a manageable number of attributes. The corresponding concept lattice is too large for an expert to comprehend. In addition, the expert might not be interested in the formal concepts from this concept lattice. Rather, the expert might want to consider aggregates of the individual respondents as objects in the formal context with the aggregates defined by having the same attributes on a set S of attributes specified by an expert, such as those regarding age, sex, etc., with S being a subset of the set Y of all attributes. Attributes from S will be called characteristic attributes.

The aggregates we consider are equivalence classes of individual respondents. For respondents $x_1, x_2 \in X$, put

$$x_1 \equiv_S x_2 \text{ if and only if } \{x_1\}^\uparrow \cap S = \{x_2\}^\uparrow \cap S.$$

Clearly, $\equiv_S$ is an equivalence relation on X and $x_1 \equiv_S x_2$ means that x_1 and x_2 have the same attributes from S, i.e. are indistinguishable by the attributes from S. We call the classes $[x]_{\equiv_S}$ of $\equiv_S$ aggregate objects and denote, furthermore,

– by X_1 the set of all classes of $\equiv_S$, i.e. $X_1 = X/\equiv_S$, by Y_1 the set of those attributes from Y not included in S, i.e. $Y_1 = Y - S$.

Now, for each class $[x]_{\equiv_S}$ from X_1 and each attribute $y \in Y_1$, we consider the relative frequency of objects in having attribute y and denote it by $I_1([x]_{\equiv_S}, y)$ or simply by $I_1(x, y)$. That is, we put

$$I_1(x, y) = \frac{|\{x_1 \in [x]_{\equiv_S} : x_1 \text{ has } y\}|}{|[x]_{\equiv_S}|}$$

We can consider I_1 a fuzzy relation which will indeed be the case in this study. Namely, we will consider a particular concept lattice associated to $\langle X_1, Y_1, I_1\rangle$, called a lattice of crisply generated fuzzy concepts [4]. For technical reasons, we round the degrees assigned by I_1 to those from the scale $\{0, 0.01, \ldots, 0.99, 1\}$.

Remark 1. An important remark is in order. Interpreting degrees of membership of a fuzzy relation by relative frequencies is not typical. It may even seem not appropriate because degrees in fuzzy logic are typically degrees associated to graded collections and relationships. However, as we will see, the way we use the degrees with frequential interpretation is all right from the semantical point of view. In fact, the only thing which matters in our manipulation with the degrees is a comparison of the degrees, i.e. a comparison of relative frequencies.

In addition, the expert might want not to include aggregate objects $[x]_{\equiv_S}$ which contain less than m objects with m being prescribed by an expert (this requirement was suggested by our domain expert).

Example 1. Consider the following illustrative example. Let the ordinary formal context be given by Tab. 1. Consider S = {SEXMale, SEXFemale, JOByes, JOBno} as the set of characteristic attributes.

Using the above-described transformation, we obtain a formal fuzzy context with 4 aggregate objects Fno (women who do not have a job), Fyes (women who have a job), Mno (men who do not have a job), Myes (men who have a job), depicted in Tab. 2.

Table 1. Original context.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0
2	1	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0
3	1	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
4	1	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
5	1	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
6	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
7	1	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0
8	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0
9	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0
10	0	1	1	0	1	0	0	0	0	1	0	0	0	1	0	0
11	0	1	1	0	0	0	1	0	0	0	1	0	0	1	0	0
12	0	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0
13	0	1	1	0	1	0	0	0	0	1	0	0	0	1	0	0
14	0	1	1	0	1	0	0	0	0	1	0	0	0	1	0	0
15	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0
16	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0	0
17	0	1	0	1	1	0	0	0	0	1	0	0	0	1	0	0
18	0	1	0	1	1	0	0	0	0	1	0	0	0	1	0	0
19	0	1	0	1	1	0	0	0	0	1	0	0	0	1	0	0

attributes:1 - SEXMale, 2 - SEXFemale, 3 - JOByes, 4 - JOBno, 5 - AGE15-24, 6 - AGE25–34, 7 - AGE35–44, 8 - AGE45–54, 9 - AGE55–65, 10 - BMIunder (Body Mass Index =body weight $[kg]$ / body height $[m]^2$), 11 - BMInormal, 12 - BMIover, 13 - BMIobesity, 14 - PAlow, 15 - PAmoderate, 16 - PAhigh

Table 2. Fuzzy context derived from the formal context from Tab. 1.

	1	2	3	4	5	6	7	8	9	10	11	12
Fno	0.75	0	0	0	0.25	1	0	0	0	1	0	0
Fyes	0.67	0.17	0.17	0	0	0.67	0.33	0	0	1	0	0
Mno	0.25	0	0.25	0.5	0	0.25	0.75	0	0	1	0	0
Myes	0.20	0	0	0.80	0	0	0.80	0.20	0	1	0	0

attributes: 1 - AGE15-24, 2 - AGE25-34, 3 - AGE35-44, 4 - AGE45-54, 5 - AGE55-65, 6 - BMIunder, 7 - BMInormal, 8 - BMIover, 9 - BMIobesity, 10 - PAlow, 11 - PAmoderate, 12 - PAhigh

3 Concept lattices from the derived contexts with relative frequencies

Given a formal fuzzy context $\langle X_1, Y_1, I_1\rangle$ consisting of a set X_1 of aggregate objects, a set Y_1 of their attributes, and a fuzzy relation I_1 with $I_1([x]_{\equiv_S}, y)$ representing relative frequency of attribute y in class $[x]_{\equiv_S}$, we use so-called crisply generated fuzzy concept lattice $\mathcal{B}(X_1, Y_1, I_1)$ for displaying the information contained in the questionnaire. Due to the limited scope, we refer the reader to [4] for a detailed information on crisply generated concept lattices.

Let us recall that a crisply generated concept lattice is a part of a fuzzy concept lattice [2, 15] generated from the arrow operators by crisp (i.e. ordinary) sets of objects (or dually, attributes). In a fuzzy setting, the deriving arrow operators are defined as follows. Let L be a scale of truth degrees such as our $L = \{0, 0.01, \ldots, 1\}$. For fuzzy sets $A \in L^{X_1}$ of objects and $B \in L^{Y_1}$ of attributes, define sets $A^{\uparrow} \in L^{Y_1}$ of attributes and $B^{\downarrow} \in L^{X_1}$ of objects by

$$A^{\uparrow}(y) = \bigwedge_{[x]_{\equiv_S} \in X_1} (A([x]_{\equiv_S}) \to I_1([x]_{\equiv_S}, y)),$$

$$B^{\downarrow}([x]_{\equiv_S}) = \bigwedge_{y \in Y_1} (B(y) \to I_1([x]_{\equiv_S}, y)).$$

Here, $\to$ is a suitable residuum (corresponds to implication connective). Note, however, that for the crisply generated formal concepts which are of interest in our approach, it does not matter, which residuum $\to$ one takes. Namely, for all the residua, the corresponding crisply generated concept lattices are isomorphic, see [4]. Now, a crisply generated fuzzy concept lattice $\mathcal{B}(X_1, Y_1, I_1)$ is the set of all fixpoints of the arrow operators which are generated by a crisp set of objects, i.e.

$$\mathcal{B}(X_1, Y_1, I_1) = \{\langle A, B\rangle \mid A^{\uparrow} = B, B^{\downarrow} = A, \text{ and } A = C^{\uparrow\downarrow} \text{ for some crisp } C \in L^{X_1}\}.$$

Note that C being crisp means that $C([x]_{\equiv_S}) = 0$ or $C([x]_{\equiv_S}) = 1$ for each $[x]_{\equiv_S} \in X_1$.

Remark 2. Formal concepts $\langle A, B\rangle$ from $\mathcal{B}(X_1, Y_1, I_1)$ can be interpreted as follows. A is a collection of aggregate objects (i.e. classes of individual respondents). B is a fuzzy set such that every aggregate object (class of individual respondents) from A has attribute y with relative frequency at least $B(y)$. Strictly speaking, A is a fuzzy set, too, and we should say that $\{[x]_{\equiv_S} \mid A([x]_{\equiv_S}) = 1\}$ is the set of aggregate objects such that for B, every $[x]_{\equiv_S} \in A$ has attribute y with relative frequency at least $B(y)$. For aggregate objects $[x]_{\equiv_S}$ with $A([x]_{\equiv_S}) < 1$, high $A([x]_{\equiv_S})$ indicate that the relative frequencies of y's are close to $B(y)$.

Note that crisply generated fuzzy concept lattices are isomorphic to Krajči's one-sided fuzzy concept lattices defined in [14] as well as to the fuzzy concept lattices defined in [7].

4 Evaluation of the questionnaire

This concept lattice $\mathcal{B}(X_1, Y_1, I_1)$ is suitable for further analysis. A frequent expert request is to find out common properties for groups of respondents (i.e. aggregate objects) which we represent by characteristic attributes. For this purpose, we find the smallest concept, which include all request aggregate objects (in degree 1). For the expert, it is important how many and which respondents are contained in such a concept. The number of respondents is sum of the numbers of objects contained in the aggregate objects which are present in the extent of the formal concept.

No less important are characteristics of respondents represented by the concept. This is given by characteristic attributes, which characterize aggregate objects included in this concept in degree 1. For example, if there is an aggregate object in the extent corresponding to attributes SEXFemale and JOBno, and and aggregate object corresponding to attributes SEXMale and JOByes, we can see, that this concept relates to women who do not have a job and simultaneously to men who do have a job.

Another type of information is contained in the intent B. A degree $B(y)$ expresses the precentage of objects which are present in the extent (in degree 1) and have attribute y. Both high and low values of $B(y)$ are interesting for the expert. For example the concept whose extent is described above and which has in its intent attribute SMOKINGno to degree 0.42 and attribute DOGyes to degree 0.65, we can see, that at least 42% of women without a job and working men do not smoke and at least 65% of women without a job and working men have a dog.

By passing down the concept lattice we can examine common attributes of different subsets of aggregate objects. A comparison of concepts created from aggregate objects is also interesting for the expert. For example, if the expert is about to examine the influence of physical activity on population, then comparing a concept which is characterized by aggregate objects SEXMale and PAlow, and SEXFemale and PAlow, with a concept, which is characterized by aggregate objects SEXMale and PAhigh, and SEXFemale and PAhigh is of interest.

5 IPAQ

International Physical Activity Questionnaire (IPAQ) is an appropriate instrument with acceptable measurement properties for monitoring of physical activity (PA) among 18 to 65 year old adults in diverse (international) populations ([1], [20]).

Every participant is filling out the time spent on performing physical activities including walking, including personal data (sex, age, body weight and height) and lifestyle characteristics (education, job time, type of housing - own house apartment, family status - living alone $\times$ in family $\times$ in family with children, smoking - yes $\times$ no, ownership of a car, a dog, a bike or weekend house, participation in an organized PA). The most important result from the IPAQ is the finding how many adults (%) meet the category "low", "moderate" or "high" level of PA in relation to the guidelines of healthy lifestyle and health support. However, complex and transparent description of groups of adults with different PA level is necessary. This description should include environmental, social, and somatic factors determining the lifestyle of the participants. Simple and multiple correlation analysis did not show ($-0.15 \leq r_S \leq +0.15$) deeper associations between PA level and personal data and lifestyle characteristics of participated adults. Therefore, it is not possible to say that smokers or obese adults are less physically active than non-smokers or non-obese adults, due to very low correlations between PA level and smoking or body weight. This is the reason why we are looking for a "suitable tool" that will allow complex and transparent characterization of groups of adults with regard to their PA level and lifestyle characteristics. Formal concept analysis proved useful for this purpose [16].

Are there any differences between groups of females with low and high level of PA in relation to the guidelines of healthy lifestyle and health support? Are there any similarities in personal data or lifestyle characteristics in these two groups of females? Are any of these characteristics more typical for group of females with high level of PA than for group of females with low PA level? These are the types of questions formal concept analysis helps the expert with in looking for answers. The whole questionnaire data was transformed into an ordinary formal context that included 4510 objects (participants) and 47 attributes. The next step was to create agregate objects. The set of characteristic attributes contains attributes SEXMale, SEXFemale, PAlow, PAmoderate and PAhigh. As explained earlier, this way we come to a fuzzy context with 6 aggregate objects and 42 attributes. The aggregate objects are MLo (men with low PA), MMo (men with moderate PA), MHi (men with high PA), FLo (women with low PA), FMo (women with moderate PA), FHi (women with high PA). The resulting fuzzy concept lattice contains 54 concepts.

Evaluation of this concept lattice helps answer the questions formulated above. If the expert is interested in data related to women, he/she will work with the part of the concept lattice depicted in Fig. 1. This part contains formal concepts with extents with aggregate objects FLo, FMo, and FHi in degree 1. For instance, there is obviously high number of females who showed at least moderate PA level. This result is strongly influenced by everyday walking (50% of female spend more than 65 minutes per day). Ownership of a car or a dog, living in house, living alone, smoking and participation in an organized PA more than 3 times per week are attributes that do not affect the level of PA (Tab. 3). Females characterized as high active are more walking and have more optimal body weight (BMI index). We can also see that more females with low level of PA have a job. Different value of attribute Age 15-24 can indicate a fact that physical activity of young people is smaller (Tab. 3).

On the other hand we can be interested in finding features that are shared by more aggregate objects. For example, what is common for women and men with high physical activity. In this case we can examine the intent of a concept whose extent contains aggregate objects MHi and FHi in degree 1 (Tab. 4).

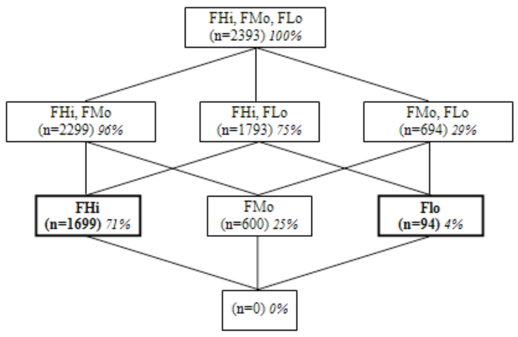


Fig. 1. The set of concepts of females differentiated according to health supported PA level, n=sum of cardinality of objects in extent in degree 1, $x\% = (n/N) * 100$ where N is sum of cardinality FLo, FMo and FHi, ie. number of woman in questionnaire

Table 3. Differences between high and low active group of females

	1	2	3	4	5	6	7	8	9	10	11	12	13
FHi (n=1699)	0.51	0.08	0.64	0.21	0.43	0.08	0.29	0.52	0.43	0.17	0.30	0.65	0.40
FLo (n=94)	0.52	0.09	0.61	0.18	0.40	0.10	0.24	0.60	0.36	0.31	0.10	0.78	0.52

attributes: 1 - Living in house, 2 - Living alone, 3 - Ownership of a car, 4 - Smoking, 5 - Ownership of a dog, 6 - Organized PA ≥ 3 per week, 7 - Ownership of a weekend house, 8 - BMI under, 9 - BMI normal, 10 - Walking low, 11 - Walking middle, 12 - Job yes, 13 - Age 15-24

Table 4. Attributes shared by women and men with high PA

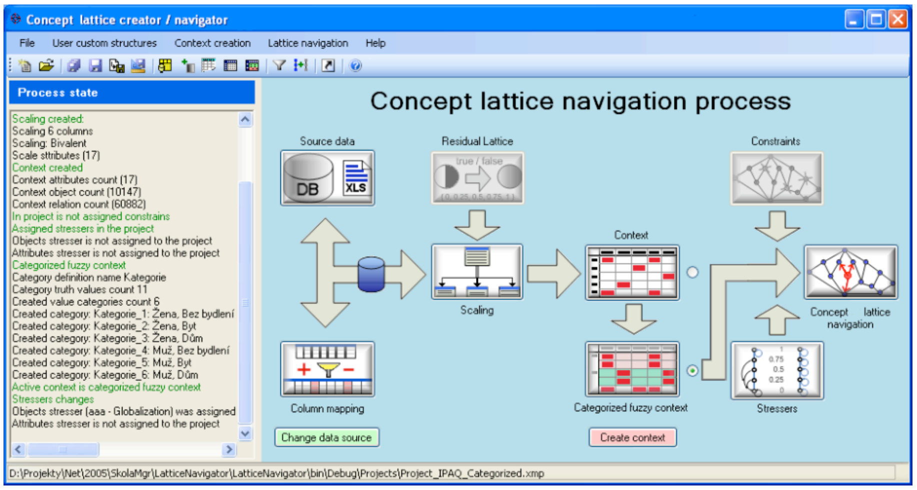
	1	2	3	4	5	6	7	8	9	10	11	12
FHi,MHi	0.84	0.64	0.65	0.21	0.37	0.49	0.52	0.17	0.09	0.43	0.04	0.01

attributes: 1 - Ownership of a bicycle, 2 - Ownership of a car, 3 - Job yes, 4 - Smoking, 5 - Ownership of a dog, 6 - Living in house, 7 - Walking high, 8 - Walking low, 9 - BMI under, 10 - BMI normal, 11 - BMI over, 12 - BMI obesity

These are examples of how our approach via formal concept analysis assists the expert in evaluating the questionnaire data. A detailed description of the interpretation of the data is beyond the scope of this paper.

6 Software tool

We used a software tool which is developed in the Department of Computer Science at Palacky University, Olomouc, to create the fuzzy context and to browse the corresponding fuzzy concept lattice. This software tool supports the whole process of the processing and evaluation of IPAQ questionnaire. The basic overview of functions that are supported and their succession is in Fig. 2.



```
Scaling created:
Scaling 6 columns
Scaling: Bivalent
Scale sttributes (17)
Context created
Context attributes count (17)
Context object count (10147)
Context relation count (60882)
In project is not assigned constrains
Assigned stressers in the project
Objects stresser is not assigned to the project
Attributes stresser is not assigned to the project
Categorized fuzzy context
Category definition name Kategorie
Category truth values count 11
Created value categories count 6
Created category: Kategorie_1: Žena, Bez bydlení
Created category: Kategorie_2: Žena, Byt
Created category: Kategorie_3: Žena, Dům
Created category: Kategorie_4: Muž, Bez bydlení
Created category: Kategorie_5: Muž, Byt
Created category: Kategorie_6: Muž, Dům
Active context is categorized fuzzy context
Stressers changes
Objects stresser (aaa - Globalization) was assigned
Attributes stresser is not assigned to the project
```

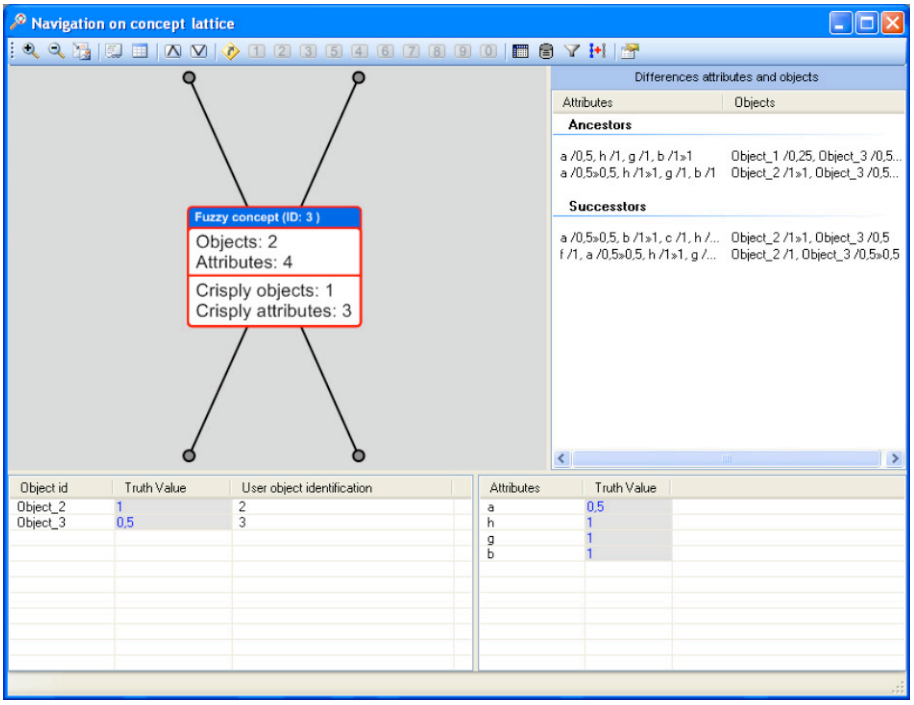
```
D:\Projekty\Net\2005\SkolaMgr\LatticeNavigator\LatticeNavigator\bin\Debug\Projects\Project_IPAQ_Categorized.xmp
```

Fig. 2. Base screen of application

The processing of the questionnaire consists of the following steps.

- **Reading of data**. IPAQ questionnaire is recorded in the form of MS Excel file. The columns of this file contain respondents' answers to individual questions. The software tool allows to specify which columns are included in the processing.
- **Scaling**. The answers to some questions may be in the form of many-valued attributes. For example, the values in the column Age may be in the interval from 18 to 69. Due to this fact it is necessary to transform the original file to the form in which each column contains only 0 or 1. This process is called scaling. Our software tool allows one to specify the bivalent attributes and the scale for each column in data source file.
- **Creating aggregate objects**. It is possible to interactively specify the set of characteristic attributes. The user also chooses the residuated lattice in this step including the number of truth degrees.

The fuzzy context is created after these steps. We can examine the fuzzy concept lattice and its concepts. Our software tool does not create the concept lattice as a whole. Instead, it supports an interactive navigation in the concept lattice. It shows the information related to the current concept and its direct neighbors. A user selects next steps by choosing an ancestor or successor of the current concept. He/she can move from a more general concept to a more special concept and vice versa. He/she can also specify the content of the extent or the intent and move to the appropriate concept. We can see the user's screen in Fig. 3.



Attributes	Objects
Ancestors	
a /0,5, h /1, g /1, b /1»1	Object_1 /0,25, Object_3 /0,5...
a /0,5»0,5, h /1»1, g /1, b /1	Object_2 /1»1, Object_3 /0,5...
Successtors	
a /0,5»0,5, b /1»1, c /1, h /...	Object_2 /1»1, Object_3 /0,5
f /1, a /0,5»0,5, h /1»1, g /...	Object_2 /1, Object_3 /0,5»0,5

Object id	Truth Value	User object identification
Object_2	1	2
Object_3	0,5	3

Attributes	Truth Value
a	0,5
h	1
g	1
b	1

Fig. 3. Navigation in fuzzy concept lattice

The navigation in the concept lattice needs the calculation of the current concept and its neighbors only. This calculation is relatively fast and does not depend on the size of the whole concept lattice. Due to this fact the navigation proceeds on-line and the user can modify the course of navigation interactively, based on information gained. The user can also specify additional constraints to be satisfied by formal concepts which are to be presented to him/her.

7 Conclusions

We presented a way formal concept analysis can help in evaluation of questionnaires. The proposed method has been tested on evaluation of IPAQ questionnaire. The method provides an expert with a structures view on the questionnaire data with some elementary statistics and enables the expert to see important relationships in the data.

References

1. Bauman A., Chey T., Bowles H., Smith B., Meron D., Ainsworth B., Jones D. A., Craig C., Cameron C., Sjostrom M., Hagestromer M., Fromel K., Mitas J. *et al.*: *International physical activity prevalence estimates: Results from the International Prevalence Study in 20 Countries*, Medicine and Science in Sports and Excercise (in press).
2. Belohlavek R.: *Fuzzy Relational Systems: Foundations and Principles.* Kluwer, Academic/Plenum Publishers, New York, 2002.
3. Belohlavek R.: Concept lattices and order in fuzzy logic. *Annals of Pure and Applied Logic* **128**(1–3)(2004), 277–298.
4. Belohlavek R., Sklenar V., Zacpal J.: Crisply generated fuzzy concepts. ICFCA 2005, Int. Conf. Formal Concept Analysis, *LNAI* 3403, pp. 268–283, Springer-Verlag, Berlin/Heidelberg.
5. Belohlavek R., Vychodil V.: Reducing the size of fuzzy concept lattices by hedges. In: FUZZ-IEEE 2005, The IEEE International Conference on Fuzzy Systems, May 22–25, Reno (Nevada, USA), pp. 663–668.
6. Belohlavek R., Vychodil V.: What is a fuzzy concept lattice? In: Proc. CLA 2005, 3rd Int. Conference on Concept Lattices and Their Applications, September 7–9, Olomouc, Czech Republic, pp. 34–45.
7. Ben Yahia S., Jaoua A.: Discovering knowledge from fuzzy concept lattice. In: Kandel A., Last M., Bunke H. (Ed.): *Data Mining and Computational Intelligence*, pp. 167–190, Physica-Verlag.
8. Carpineto C., Romano G.: *Concept Data Analysis. Theory and Applications.* J. Wiley, 2004.
9. Craig C. L. et al.: *International physical activity questionnaire: 12-country reliability and validity*, Medicine and Science in Sports and Excercise, 35, 1381–1395, 2003.
10. Frömel K. *et al.*: *Pohybová aktivita a inaktivita obyvatel České republiky v kontextu behaviorálnch změn*, RP: 6198959221, Olomouc, Univerzita Palackého, 2005.
11. Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.
12. Hájek P.: *Metamathematics of Fuzzy Logic.* Kluwer, Dordrecht, 1998.
13. Klir G. J., Yuan B.: *Fuzzy Sets and Fuzzy Logic. Theory and Applications.* Prentice-Hall, 1995.
14. Krajči S.: A generalized concept lattice. *Logic J. of IGPL* **13**, 543–550.
15. Pollandt S.: *Fuzzy Begriffe.* Springer-Verlag, Berlin/Heidelberg, 1997.
16. Sigmund E., Zacpal J., Sklenář V., Frömel K.: *New Approach To Evaluation Of IPAQ Questionnaire Physical Aktivity Data: Formal Concept Analysis.* Tudory, 4th International Conference Movement and Health, p. 173, ISBN 80–244–1166–0, Palacky University, 2005.

17. Sklenář V., Zacpal J., Sigmund E.: *Evaluation of IPAQ questionnaire by FCA*, CLA 2005, pp. 60–69, ISBN: 80–248–0863–3, Palacky University, Olomouc, 2005.
18. Thomas J. R., Nelson J. K., Silverman S. J.: *Research Methods in Physical Activity.* Human Kinetic, Champaign, 2005.
19. Wille R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (Ed.): Ordered Sets, 445–470, Reidel, Dordrecht-Boston, 1982.
20. http://www.ipaq.ki.se/

Camelis: Organizing and Browsing a Personal Photo Collection with a Logical Information System

Sébastien Ferré

Irisa/Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex, France
Email: ferre@irisa.fr

Abstract Since the arrival of digital cameras, many people are faced to the challenge of organizing and retrieving the overwhelming flow of photos their life produces. Most people put no metadata on their photos, and we believe this is because existing tools make a very limited use of them. We present a tool, Camelis, that offers users with an organization of photos that is dynamically computed from the metadata, making worthwhile the effort to produce it. Camelis is designed along the lines of Logical Information Systems (LIS), which are founded on logical concept analysis. Hence, (1) an expressive language can be used to describe photos and query the collection, (2) manual and automatic metadata can be smoothly integrated, and (3) expressive querying and flexible navigation can be mixed in a same search and in any order. This presentation is illustrated by experiences on a real collection of more than 5000 photos.

1 Introduction

Formal Concept Analysis (FCA) has been recognized as a good paradigm for information retrieval [GMA93,ML02] because it makes it possible to tightly combine querying and navigation in a same search. Querying alone is not satisfying because it requires from users to know the query language, and to have a precise idea of what they search for. Navigation, by leading users to the result step by step, is more interactive, but the navigation structure is most often very rigid so that only one or a few paths exist to each object (e.g., file hierarchy, hyperlink graph). In FCA, the concept lattice plays the role of the navigation structure. Each concept combines a query as a set of attributes (the intent), and a navigation place as a set of objects (the extent). Attributes can be added to the query in any order, so that a concept can be reached through several paths.

Logical Information Systems (LIS) [FR04] have been introduced (1) to combine querying and navigation, (2) to be reasonably efficient on large collections of objects, (3) to make use of an expressive language for object descriptions and queries, and (4) to be generic w.r.t. the kind of objects and the language. Because of (3) it becomes necessary to do complex reasoning to see whether an object description match a query (e.g., the object description is a string, and the query is a regular expression). Logics are the right things to encapsulate representation and reasoning facilities. So we defined a generalization of FCA, Logical Concept Analysis (LCA) [FR00], where logical formulas can be used instead of sets of attributes. This makes FCA an instance of LCA, where object descriptions and queries are sets of attributes.

Camelis[1] is a complete implementation of a logical information system. It is generic in that a logic module can be plugged in so as to cover different application needs. It uses specific data structures and algorithms so that it is efficient up to 10,000 objects. It has a graphical interface that displays at all time the query that led to the current concept, its extent, and properties to be added or removed from the current query in order to reach neighbour concepts. Both browsing and defining the context are possible through this interface.

Among the various existing applications of Camelis, the most convincing is the management of a photo collection. Indeed, photos can be described along many facets like date, location, event, visible persons, visible objects, etc. A file hierarchy enforces a strict order between these facets, making some search hardly possible. Tag-based systems like Flickr are limited because a photo tagged with 'Sydney' as location will not be answer to a query containing 'Australia'; and a photo tagged with 'formal concept analysis' will not be answer to '...concept analysis'. These limitations are easily solved by dedicated logics: here, a taxonomy of locations, and a logic of string patterns. In this paper, we illustrate the capabilities of Camelis and LIS on a real context, the personal photo collection of the author. It contains more than 5,000 photos, and has been incrementally defined since 2003, along the arrival of new photos.

Section 2 assumes an existing context, and presents all the facilities provided by Camelis to browse and retrieve photos, from querying by formulas and querying by objects, to different kinds of navigation: i.e., downward and upward, backward and forward, and sideward. Section 3 illustrates the incremental definition of the context with the arrival of a new pack of photos. Section 4 compares Camelis with related tools.

2 Retrieving Photos

In this section, we assume the existence of a logical context representing a collection of photos and their metadata. The process of building such a context is developed in Section 3. We recall the basics of Logical Concept Analysis (LCA) [FR00,FR04], whose principles are the same as in FCA, except that *logical properties* partially ordered by a *subsumption* relation are manipulated instead of unrelated attributes.

Definition 1 (logical context). *Let $\mathcal{L} = (L, \sqsubseteq)$ be a* logic*, i.e. a set of properties L partially ordered by a* subsumption *relation $\sqsubseteq$. A* logical context *is a*

[1] See `http://www.irisa.fr/LIS/ferre/camelis/`.

triple $K = (\mathcal{O}, \mathcal{L}, D)$, where $\mathcal{O}$ is a set of object identifiers, and D is a mapping from objects to their description as a set of logical properties.

In our application, object identifiers are file paths or URLs to photos, and their logical description is a set of properties such as location, date, event, etc. These properties are either taxonomic terms or valued attributes over various concrete domains such as date intervals or string patterns. The subsumption ordering is fixed for concrete domains, and designed by hand for taxonomies.

– location: `Montpellier` $\sqsubseteq$ `France` $\sqsubseteq$ `'European Union'`
– date: `date = 24 oct 2007` $\sqsubseteq$ `date in sep 2007 .. jun 2008`
– event: `event is "conference CLA"` $\sqsubseteq$ `event contains "conference"`

An object can be in the extent of a property without having it explicitely in its description, through subsumption.

Definition 2 (extent). *Let $K = (\mathcal{O}, \mathcal{L}, D)$ be a logical context, and $p \in \mathcal{L}$ a property. The* extent *of property p is defined by the set of objects whose description entails p:*

$$extent(p) = \{o \in \mathcal{O} \mid \exists d \in D(o) : d \sqsubseteq p\}.$$

Hence, object descriptions can be kept small and precise at the same time. For instance, an object described with some date d will be in the extent of any date interval containing d. Here lies the main benefit of logics w.r.t. attributes. A similar benefit can be obtained with conceptual scaling [GW99], but only for finite domains (e.g., locations), and with redundancy in the scaled context [CFRD06].

Logical properties can be combined by boolean operators *and*, *or*, *not* so as to form more complex *queries*. Their extent is defined inductively as follows for every queries $q1, q2$:

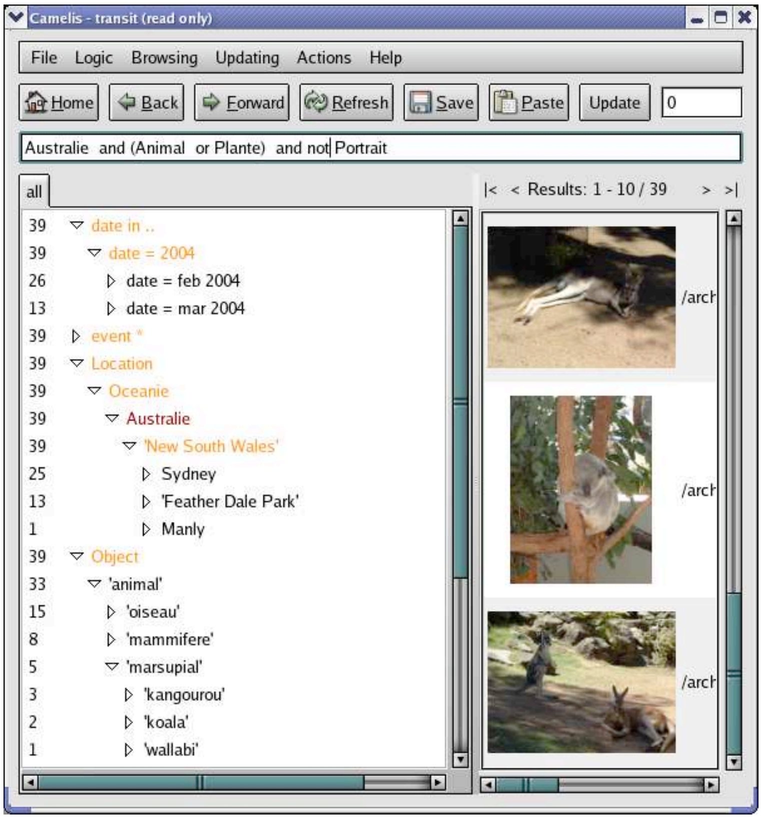
$$\begin{aligned} extent(q1\ and\ q2) &= extent(q1) \cap extent(q2) \\ extent(q1\ or\ q2) &= extent(q1) \cup extent(q2) \\ extent(not\ q1) &= \mathcal{O} \setminus extent(q1) \end{aligned}$$

This definition makes every pair $(q, extent(q))$ an inf-semiconcept of a logical context whose logic is extended with boolean operators. These semi-concepts play the role of both query and navigation place in the following, founding the combination of querying and navigation presented in the introduction.

In the next sections we present the many different ways information can be retrieved in Camelis. This is illustrated on a logical context made of 5,480 photos described by an average of 10 properties each, taken among a set of 28,143. This later figure is not the number of *available* properties, which is infinite, but the number of properties *used* so far.

2.1 The Graphical Interface

The graphical interface of Camelis can be seen in Figure 1. At every time, the *current query* and its *extent* are displayed respectively as a text field at the top


Camelis - transit (read only)
File Logic Browsing Updating Actions Help
Home Back Forward Refresh Save Paste Update
all
|< < Results: 1 - 10 / 39 > >|
/arch
/arch
/arch

```
0
Australie  and (Animal  or Plante)  and not Portrait

39   ▽ date in ..
39     ▽ date = 2004
26       ▷ date = feb 2004
13       ▷ date = mar 2004
39   ▷ event *
39   ▽ Location
39     ▽ Oceanie
39       ▽ Australie
39         ▽ 'New South Wales'
25           ▷ Sydney
13           ▷ 'Feather Dale Park'
1            ▷ Manly
39   ▽ Object
33     ▽ 'animal'
15       ▷ 'oiseau'
8        ▷ 'mammifere'
5        ▽ 'marsupial'
3          ▷ 'kangourou'
2          ▷ 'koala'
1          ▷ 'wallabi'
```

Figure1. A screenshot of the graphical interface of Camelis.

and as a thumbnail list at the right. The query is editable, and the extent can be scrolled page by page (the page size can be customized). At the left is displayed a set of property trees, where the hierarchical relations match the subsumption relations between logical properties. As a logic can be any partial ordering, not necessarily a tree, the same property can appear several times. This is not a problem as tree nodes are expanded on demand.

These trees provide a feedback about the current extent, and a support for navigation. Each visible property has a *count* (numbers at the left), and a color that depends on the context and the current query. Let K be a logical context, q be the current query, and p be a property. The count of p is the number of objects in the current extent that have p as a property:

$$count(p) = |extent(q) \cap extent(p)|.$$

A property is not shown if its count is 0, because it tells nothing about the current extent. In order to get fewer properties in the property trees it is possible to set a minimum count on each property to control the way it is expanded. A property is red- or orange-colored if it is shared by all objects in the extent, i.e. if its count is equal to the size of the extent. This means the set of orange/red properties is the intent associated to the current extent, and red properties are those properties also occuring in the query. Because the count of properties depends on the current extent, property trees need to be recomputed each time the query changes. This makes the navigation trees very dynamic and informative. Specific data structures are used to make their computation efficient, i.e. linear in the size of the context (see [FR04,PR05] for data structures and algorithms). The following sections show how these informations can be used to navigate among photos along various directions, and to get feedback about selected extents.

2.2 Navigating Downward in the Concept Lattice

Firstly, suppose I[2] want to find some photos from a trip in Australia for ICFCA'04. I first expand the property `Location`, and find I have photos from Europe (4859), Africa (162), and Australia (148). After selecting the property `Australia`[3]:

- the query becomes `Australia`,
- the property `Australia` becomes red because it is now part of the query, and it is automatically expanded to show sub-locations of Australia,
- the properties `Europe` and `Africa` are no more visible, because no more relevant (count = 0),
- the extent displays the first page of the 148 selected photos.

The sub-location `'New South Wales'` becomes orange, which means this is the only region of Australia where I have taken photos. More precisely I find that I have been mainly in Sydney (105), and in the Blue Mountains (18).

Now I expand the property `Type` and see there are different types of photos: e.g., buildings (29), animals (34), plants (6). I get interested in Australian living things, so I select both `Animal` and `Plant`, which leads me to the refined query `Australia and (Animal or Plant)`, whose extent contains 40 photos. One of these photos is a portrait, which I do not want, so I select the negation of `Portrait` with the help of a contextual menu. This leads me to the new query `Australia and (Animal or Plant) and not Portrait` (39 photos). By expanding more properties, I discover that these photos were taken in February and March 2004 in New South Wales, that 5 photos of 3 different species of marsupials are present: e.g., kangooroo, koala, wallabi.

These 3 navigation steps led to semi-concepts with more and more precise queries, and hence smaller and smaller extents. This is called *downward* navigation, and its principle is to combine with a `and` the current query and a refinement. A refinement can be a single property, a disjunction of properties,

[2] The pronoun I is used to emphasize a personal experience in the use of CAMELIS.

[3] French words can be seen in screenshots because it is my real photo collection, but english translations are used in the text for better understanding.

or the negation of a property. When a refinement uses only properties that are relevant and not colored, the new semi-concept is always strictly smaller but not empty. This is a big advantage compared to purely querying systems, where it is common to get empty results.

Figure 1 shows the interface obtained after the previous navigation operations. At this stage, I can either browse the 39 photos in Camelis, or launch a diaporama in an external application.

2.3 Navigating Backward and Forward in the Query History

Like in web browsers, it is possible to navigate *backward* and *forward* in the query history. This is mainly useful for opening parentheses on the path of a navigation. Imagine that, while browsing the extent at the end of previous section, I find a photo of a koala, and want to look at all other photos of koala. I first select the property `'koala'`, which leads me to a new semi-concept with 2 photos, and then I can move back to the previous semi-concept. When moving back, scrolling positions are remembered, so that I can go on easily in my browsing of Australian animals and plants.

2.4 Navigating Upward in the Concept Lattice

During downward navigation, I sometimes want to remove or generalize some properties in the query so as to get larger extents: this is *upward* navigation. For instance, I realise I have not enough photos of animals and plants. If I want to remove the last refinement, the back button is a simple way to achieve this. But if I want to remove the first refinement `Australia`, I would need to move 3 steps backward, and re-select the last 2 refinements. I could also edit the query by hand, but users usually prefer to navigate rather than editing queries.

Besides, orange and red properties are shared by all extent objects, and so cannot be used for downward navigation. This makes them available for upward navigation. When a red property is selected, it is removed from the query. For instance, if I select `Australia`, the new query is `(Animal or Plant) and not Portrait` (282 photos from many locations). When an orange property is selected, it replaces more specific properties in the query. For instance, if I select `Pacific`, the new query is `(Animal or Plant) and not Portrait and Pacific`. In this latter case, the property `Pacific` becomes red because it is in the new query, and the property `Australia` becomes orange because this generalization gives no additional photo.

2.5 Navigating Sideward

We show in this section that downward and upward navigation can be combined in 2 forms of *sideward navigation*. From the previous query `Australia and (Animal or Plant) and not Portrait`, we first select the property `Plant` to reach the query `Australia and not Portrait and Plant` (6 photos). This is our starting point for sideward navigation.

At this point, I see that 1 photo has also the type `Landscape`, which interests me. So I select this property (downward navigation), and as the result has only 1 photo, I generalize it by removing the property `Plant` from the query (upward navigation). We have done a sidestep from Australian plants (6 photos) to Australian landscapes (80 photos), replacing in the query the property `Plant` by the property `Landscape`. From there, I perform a new sidestep from the property `Landscape` to the property `Building`, now watching 28 photos of Australian buildings. These steps are suggested and supported by photos sharing two properties. This illustrates the relevance of assigning several types to photos, which is very common in this photo context. The same would apply to persons visible on photos, as a photo often shows several persons.

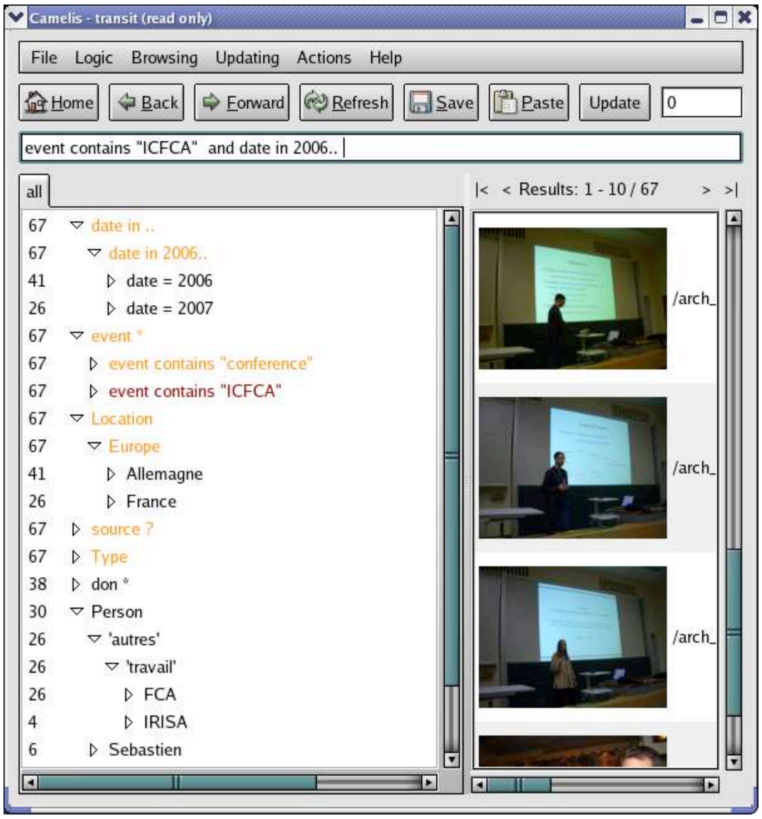
The same does not apply to locations, as a photo cannot be taken in 2 incomparable locations (e.g., in Australia and Europe). However it is still possible to navigate sideward among locations, through the taxonomy of locations. Suppose I want to find building photos from other locations. I first generalize `Australia` by `Location` in the query (upward navigation), and then browse suggested locations before selecting `Spain` (downward navigation). I thus have done a sidestep from Australia to Spain, and find 18 photos, which appear to be mainly churches taken in the north-west of Spain in 2003.

The latter form of sideward navigation is a downward-upward combination, and can be qualified as *contextual* because it relies on objects sharing some properties. The former form of sideward navigation is an upward-downward combination, and can be qualified as *logical* because it relies on subsumption relations between properties.

2.6 Querying by Formulas

Most useful queries can be reached by a succession of navigation steps, but not all. Indeed the logic allows the expression of string patterns (e.g., on events) and arbitrary intervals (e.g., on dates), and the navigation trees cannot display them all. However it is always possible to use these patterns and intervals by directly editing the query. For instance, suppose I want to find ICFCA-related photos, I enter `event contains "ICFCA"` in the query field, and find myself in the same situation as if I had selected the corresponding property in the navigation trees. I find that the 68 photos in the extent are scattered in 3 different years (2004, 2006, 2007) and in 3 different locations (Dresden, Clermont-Ferrand, Sydney), and they show people from the FCA community. I can further refine my search to photos taken since 2006 by modifying the query into `event contains "ICFCA" and date in 2006..` I now find that both year 2004, and location Sydney have disappeared from the navigation trees as they are no more relevant to the new query. The result can be seen in Figure 2.

It can be observed in Figure 2 that the 2 properties used in the query are now visible. Each time a new property is given by a user, it is inserted in the property trees for future use. Indeed, if a user has found this property useful, she or another user may well find it useful on another occasion. Now, if a user is not satisfied by this behaviour, she can still hide it, as well as any other property


Camelis - transit (read only)
File Logic Browsing Updating Actions Help
Home Back Forward Refresh Save Paste Update
all
|< < Results: 1 - 10 / 67 > >|
/arch_
/arch_
/arch_

event contains "ICFCA" and date in 2006..

67	date in ..
67	date in 2006..
41	date = 2006
26	date = 2007
67	event *
67	event contains "conference"
67	event contains "ICFCA"
67	Location
67	Europe
41	Allemagne
26	France
67	source ?
67	Type
38	don *
30	Person
26	'autres'
26	'travail'
26	FCA
4	IRISA
6	Sebastien

Figure2. Another screenshot of the graphical interface of Camelis.

by the way. Conversely it is possible to directly insert a new property, without having to use it in a query. This possibility for users to show and hide properties at will help them customizing navigation trees to their taste.

2.7 Querying by Examples

In all previous sections the query is modified either by selection of features, or by direct edition. In this section we present how a query can be determined by the selection of a subset of photos, thus supporting querying by examples. The principle is to make the query be the intent of the subset of photos.

Definition 3 (intent). *Let $K = (\mathcal{O}, \mathcal{L}, d)$ be a context, and $O \subseteq \mathcal{O}$ be a subset of objects. The* intent *of O is the set of most specific properties that are shared by all objects in O:*

$$intent(O) = Min_{\sqsubseteq}\{p \in X \mid \forall o \in O : \exists d \in D(o) : d \sqsubseteq p\},$$

where X is the set of visible properties in the property trees.

Suppose I start with the query `Australia and not Portrait`. While browsing photos in the result, I see interesting photos of buildings (e.g. 2 photos of the Opera, and 1 photo of the Harbour Bridge), and I would like to find more. By selecting them I move to a new query that is the conjunction of the properties of the intent of those 3 photos. As usual with this form of navigation, the intent query is very specific so that I get no more photos. The properties making the intent query are red in the trees, and I find some of them are very specific: e.g., a precise date (22 feb 2004), a precise location (Sydney). At this stage, I can use upward navigation to generalize the query. By removing in the query properties related to date and event, the query becomes `Sydney and Building`, and I find 29 photos. I further generalize the query by replacing the property `Sydney` by `Australia`, but find no more photos. One more generalization by removing the location, and I now find 933 photos of buildings around the world, mostly in Europe. I can also navigate downward to find photos of modern buildings, or sideward to find buildings in different countries.

A special case of querying by examples is when selecting only one photo. Then there is only one object in the extent, because there is enough properties to uniquely characterize each photo, and the query contains all the object properties, which are more easily read as red properties in the navigation trees. So this is an easy way to access the full description of any object.

3 Describing Photos

This section shows how the context that is used in Section 2 can be built in a reasonably efficient way. An important practical need is that this building can be done incrementally upon the arrival of new photos, and that everything that is done can be undone. The parts of a logical context that can be updated are (1) the set of objects (i.e., adding and removing photos), (2) the description of objects (i.e., adding and removing properties to objects), and (3) the taxonomic parts of the logic (i.e., moving a property downward and upward in a taxonomy).

3.1 Importing Photos along with Intrinsic Properties

Suppose after I took part to CLA'07, I get a new folder of photos taken during this event. In order to add these new photos to my photo context, I apply the command `Import files` to this folder so that each photo it contains becomes a new object in the context. These new objects comes with an initial description that is automatically computed from the file location and contents. The properties making this initial description are called *intrinsic*. The intrinsic properties of photos are the file location as a string, and the date. From there it is easy to select the newly imported photos by setting the query to the appropriate file location property (e.g., `file contains "My Photos/CLA2007/"`).

3.2 Adding Extrinsic Properties to Photos

The state of the art in image analysis [DLW05] makes it possible to make intrinsic low-level properties of photos, such as orientation, intensity, dominant colors or textures [ML02]. However, most high-level properties such as event or visible persons, which are the most useful, cannot be determined automatically from their contents, and have to be given manually by users [HSS02]. Those properties are called *extrinsic*. In fact there is no strict borderline between intrinsic and extrinsic properties. For instance, the location could be made instrinsic with the help of a GPS-equiped camera and a geographical information system, but these features are rarely available. Some properties (e.g., sunset) could be made instrinsic, but probably not all. The borderline is fixed as a trade-off between the cost of manually giving properties, and the cost of developping property extraction algorithms. The advantage of extrinsic properties is that they can be customized at will to the needs of users, and the design of the interface make it efficient enough as I experienced with the rich description of more than 5,000 photos.

The principle for efficiently giving new properties to photos is based on copy and paste. A set of photo thumbnails is first selected, and is then pasted on a set of properties, which can be either selected in the navigation trees, or directly entered in a text field. Removing properties is simply done by pasting on the negation of these properties. All my new photos of CLA'07 have the same event and location, so I paste all of them into `event is "conference CLA'07" and Montpellier`. Both properties are new, and are then inserted in the property trees: `event is "conference CLA'07"` is placed under `event contains "conference"`, but `Montpellier` is placed at the highest level because it is a new location. The taxonomy of locations is updated with Montpellier by the drag-and-drop of `Montpellier` under `France`, which enforces a subsumption relation between the 2 locations.

Other properties, e.g. types, persons, objects, are added in the same way. When a property already exists, it is enough to find and select it in the property trees. Otherwise it is enough to write it. In the latter case, either it is a valued attribute and it is automatically inserted, or it is a taxonomic term and it can be moved once and for all in a taxonomy. I have observed that after some number of photos I less and less often need to write new properties, and that I can rely on the property trees to maintain consistency in the use of properties. Of course the fact that I am the single user helps a lot to have a consistent vocabulary, but we could imagine a collaborative system under the principles of WIKIPEDIA or FLICKR to incrementally develop shared taxonomies.

4 Related Tools

Among applications for organizing and retrieving photos, FLICKR and PICASA are among the most famous. In PICASA, the organization of photos is limited to making albums as collections of photos. So each album can be seen as a property, and because a photo can be put in several albums, a photo can be given several properties. However there is no hierarchy between those properties, and they cannot be combined in queries: e.g., it is not possible to get the intersection, union or difference between 2 albums. Navigation is limited to the usual file system hierarchy. In Flickr, any set of properties, called tags or sets, can be given to photos, and each photo can also have a date, a geolocation, and an owner. The basic queries are conjunctions of tags, but advanced search allows for negation of tags, and an interval of dates. However, the different properties cannot all be mixed in a same query: e.g., "all photos from this author taken in this area about some subject". The navigation is limited to a cloud of tags, where the size of each tag is related to its global frequency. This would correspond to our initial navigation trees, where each tree would be limited to its root, i.e., a flat list of tags.

There also exists tools in the FCA community. Bibsonomy [HsSS06] is similar in its working and appearance to Flickr, except it applies to bookmarks and bibliographical references instead of photos. It formalizes the metadata as a triadic context that links together resources, tags, and users. ImageSleuth [DVE06] is certainly the tool the most similar to Camelis w.r.t. presentation and navigation. It displays the current semi-concept, the extent as a set of thumbnails and the intent; it provides downward and upward navigation, querying by attributes, and querying by examples. It also uses *perspectives* (sets of attributes), which are in fact simple cases of 2-levels taxonomies: the 1st level is made of perspectives like `Location` or `Person` for photos, and the 2nd level is made of attributes such as concrete locations or persons. It also provides a way to reach similar concepts according to some distance. According to the definition of this distance, we can say that our sideward navigation are a way to reach such similar concepts.

The main advantages we have compared to these tools are brought by the use of logic. Logic enables to express different kinds of properties (e.g., dates, string patterns), and to organize them according to a well-defined subsumption relation. Logic enables users to create and customize several taxonomies. Logic enables to express complex queries where all kinds of properties can be freely combined with all boolean operators. This expressiveness is nevertheless accessible through navigation as illustrated in Section 2. Another major advantage of Camelis is to provide very informative navigation trees from any query, and to support all forms of navigation and querying : navigation downward, upward, sideward, backward and forward, querying by formulas and by examples. Finally, we successfully manage a collection of photos 10 times larger than the example given for ImageSleuth, and we think this is because we do not need to compute the concept lattice. The navigation trees supporting navigation and reflecting the concept lattice are computed on the fly.

5 Conclusion

Camelis has greatly benefited from several years of application on my collection of photos. This makes it a mature implementation of Logical Information

Systems (LIS), and solves the problem of organizing and retrieving photos in a rich and flexible way. CAMELIS has also deeply changed the way I take and share photos with friends. I can quickly build customized diaporamas. For instance, to present my region Brittany to a group of foreigners, I selected all buildings and landscapes of this region, except those showing relatives. I am not reluctant to take photos that are irrelevant to the current event because I know I can easily find them afterwards: e.g., the photo of an animal during a conference event. This allows me to progressively gather collections of photos on various themes: e.g., I have got photos for 51 different species of animals, 17 different music instruments, and 255 named persons.

Besides photos, CAMELIS is also applied to music files (whose tags are automatically extracted as intrinsic properties), and to sets of bibliographical references (imported from BibTEX files and DBLP search results).

References

[CFRD06] P. Cellier, S. Ferré, O. Ridoux, and M. Ducassé. An algorithm to find frequent concepts of a formal context with taxonomy. In S. Ben Yahia and E. Mephu Nguifo, editors, *Int. Conf. Concept Lattices and Their Applications*, pages 243–248. Faculté des Sciences de Tunis, 2006.

[DLW05] R. Datta, J. Li, and J. Z. Wang. Content-based image retrieval: approaches and trends of the new age. In H. Zhang, J. Smith, and Q. Tian, editors, *Int. Work. Multimedia Information Retrieval*, pages 253–262. ACM, 2005.

[DVE06] J. Ducrou, B. Vormbrock, and P. W. Eklund. FCA-based browsing and searching of a collection of images. In H. Sch rfe, P. Hitzler, and P. Øhrstrøm, editors, *Int. Conf. Conceptual Structures*, LNCS 4068, pages 203–214. Springer, 2006.

[FR00] S. Ferré and O. Ridoux. A logical generalization of formal concept analysis. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 371–384. Springer, 2000.

[FR04] S. Ferré and O. Ridoux. An introduction to logical information systems. *Information Processing & Management*, 40(3):383–419, 2004.

[GMA93] R. Godin, R. Missaoui, and A. April. Experimental comparison of navigation in a Galois lattice with conventional information retrieval methods. *International Journal of Man-Machine Studies*, 38(5):747–767, 1993.

[GW99] B. Ganter and R. Wille. *Formal Concept Analysis — Mathematical Foundations.* Springer, 1999.

[HSS02] E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In *XLM Finland Conf.*, pages 15–27, 2002.

[HsSS06] A. Hotho, R. J schke, C. Schmitz, and G. Stumme. BibSonomy: A social bookmark and publication sharing system. In A. de Moor, S. Polovina, and H. Delugach, editors, *ICCS Work. Conceptual Structures Tool Interoperability.* Aalborg University Press, 2006.

[ML02] J. Martinez and E. Loisant. Browsing image databases with Galois' lattices. pages 791–795. ACM, 2002.

[PR05] Y. Padioleau and O. Ridoux. A parts-of-file file system. In *USENIX Annual Technical Conference, General Track (Short Paper)*, 2005.

Concept Analysis on structured, multi-valued and incomplete data

David Grosser, Henri Ralambondrainy

Laboratoire IREMIA, Université de la Réunion,
15 avenue René Cassin, BP 7151,
97715 Saint-Denis Msg. Cedex 9,
E-mail: {grosser,ralambondrainy}@univ-reunion.fr

Abstract. This paper presents an approach to Concept Analysis of structured, multivalued and incomplete data currently present in life science knowledge bases. We are concerned with tree structured objects, whose size may be variable. We focus on the composition relations between attributes in the learning process. The interest of the method is the ability to take into account both structural and value parts of the objects. An application on a coral knowledge base illustrates the advantages of the method.

1 Introduction

One of the essential issues in classification science concerns biological specimens and taxa representations and analysis [9]. This is particularly the case in marine environment for groups like corals, hydroids or sponges for which descriptions of specimens and taxonomy are particularly complex. Descriptions are often multi-valued due to variability inside of same species, structured to take into account characters dependencies and noisy or incomplete [4]. In the context of the "Knowledge Base on corals project"[5], we have developed a specific knowledge representation and analysis system: IKBS (*Iterative Knowledge based System*), to achieve identification, classification and conceptual analysis from systematic morphological descriptions.

To deal with such descriptions, we present a method for Concepts Analysis from structured, multivalued and incomplete objects. Formal Concept Analysis (FCA) [7] has been successfully applied to a range of knowledge engineering problems [14]. Traditional FCA methods and tools are usually concerned with objects described by binary contexts. Extracting concepts from more complex contexts is a recent and challenging trend of research on FCA [13]. Indeed, real-world data are often complex and difficult to be transformed in a binary format without loss of information. One key difficulty lies in the presence and management of relational attributes such as references or part-of relations between objects. For example, in [12] methods are proposed to find relational concepts in structured datasets in which individuals are described both by their own features and by their relations to other. Such data are currently found in relational or object oriented databases, or software models such as UML. In a similar research trend, [3] shows how FCA can be used to support Ontology Engineering and how ontologies can be exploited in FCA applications as background knowledge to assure consistency and scalability of the results [3].

In our approach, we are concerned with tree structured objects corresponding to specimen descriptions. The object's structure is defined in a model that represents all characteristics (attributes, relations and values) and background knowledge of a particular concept, corresponding to a taxa (family, genus, species). However, the size of each object may be different from others (different special schemas called skeletons) because of inapplicable attributes and dependencies between them. In this paper, we focus on using some background knowledge and particularly the composition relations between attributes in the Concept Analysis process. The interest of the method is the ability to take into account both structural and value part of the objects.

The paper is organized as follows: Section 2 recalls results on Galois connection on semilattices. Section 3 gives the knowledge representation model used to describe objects. Section 4 presents a way to make Concept Analysis on structured and multivalued objects. The approach is illustrated by an application example in Section 5.

2 Preliminaries

In this section, we recall some results on Galois Connection (GC) between semilattices that will be used in further sections.

Let P and Q be ordered sets. We recall that a pair $GC = (f, g)$ of maps $f : P \to Q$ and $g : Q \to P$ is a Galois Connection (GC) between P and Q if, for all $p \in P$ and $q \in Q : f(p) \leq q \iff p \leq g(q)$. The mapping $h = g \circ f$ and $k = f \circ g$ are closure operators in P and Q. Any pair (p, q) such that $(p = g(q), q = f(p))$ is called concept [7].

The definition of GC between lattices can be found in [1] and GC between semilattices has been studied by [8] [10] [6], it is useful because it gives a suitable framework for concepts analysis for data which are not binary.

We denote by O a set of objects, and Γ a meet semilattice. Let $\delta : O \to \Gamma$ be the mapping which associate every element $o \in O$ with its description $\delta(o) \in \Gamma$. The context $\mathbb{K} = (O, \Gamma, \delta)$ is called *pattern structure* in [8]. The descriptor δ induces a GC between $(\mathcal{P}(O); \subset, \cup)$ and $(\Gamma; <, \wedge)$ by means of the map, such that for $\gamma \in \Gamma$ $ext(\gamma) = \{o \in O | \gamma \leq \delta(o)\}$ and for $L \subset O$ $int(L) = \wedge_{l \in L} \delta(l)$. The GC is denoted by $GC = (ext, int)$. A concept or *pattern concept* is a pair $c = (L, \gamma)$ such that $\gamma = int(L)$ and $L = ext(\gamma)$. The subset L is called the *extension* of the concept c and γ its *intension*.

3 Knowledge representation model

The knowledge representation model is made of the descriptive model and its instances, the structured objects.

3.1 Attributes

The descriptive model represents all the observable characteristics (objects, attributes and values) pertaining to individuals belonging to a particular taxa. It is organized in a structured schema forming a tree. Each node of the tree is a component of the description defined by a list of attributes with their respective definition domain and a set of meta-data as rules, comments, hyperlinks and pictures. See Figure 1 for an overview of a descriptive model structure composed by two components, ”identification” and ”description”, itself composed by ”colony”, ”microstructure” and so on.

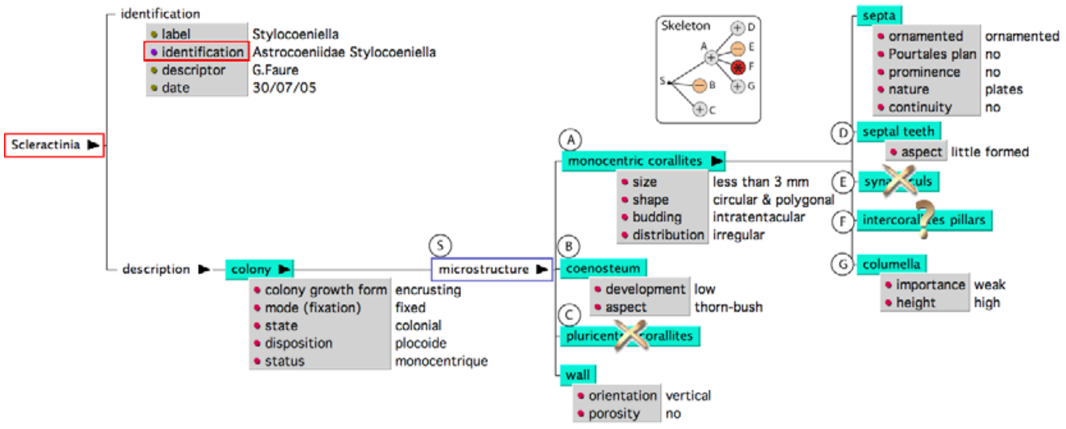


Fig. 1. Partial description of the genus *Astrocoeniidae Stylocoeniella*

Moreover, this component defines a particular boolean property, called ”possible absence”. It means that the component should necessarily be described or could be described in an object instance of the model. The attributes noted from A to G in the Figure 1 are ”contingent”, the other are ”necessary”.

Two types of attributes are considered. First, basic attributes that are usual attributes whose types are qualitative (ordinal, nominal, boolean) or quantitative (discrete or interval), and hierarchical attribute (also called taxonomic attribute) for which values are organized in a hierarchy. Second, structured attributes formed using any kind of distinct attributes.

3.2 Objects

In this section, we are concerned with the structures of objects and we shall define the meet semilattice of the structured description space of objects.

Notation We suppose given a set of basic attributes names, $A_Q = \{A_q\}_{q \in Q}$ and their corresponding domains $D_Q = \{D_q\}_{q \in Q}$. A structured attribute is recursively defined as a sequence $A :< A_1, \ldots, A_l, \ldots, A_p >$ of basic *or structured* attributes. We say that A_l is a component of A. A structured attribute is used to describe composite objects. We assume that a structured attribute A called a *schema* is given for describing a collection of objects. The set of attributes that composes A is denoted $\mathcal{A} = \{A_j\}_{j \in J}$. A structured attribute A is represented by a rooted tree $\mathcal{M} = (\mathcal{A}, \mathcal{U})$ where the set of nodes and edges are denoted by $\mathcal{A}$ and $\mathcal{U}$, respectively. The root of $\mathcal{M}$ is A, and the nodes are the attributes, basic attributes are the leaves. If $(B, B') \in \mathcal{U}$ is an edge, it means that B or B' is a component of the other.

Skeleton Let O be a set of objects described by a schema $\mathcal{M} = (\mathcal{A}, \mathcal{U})$. A *skeleton* represents the structure of an object. In the Figure 1, missing parts of the object are represented with a cross, and ? means that the component is undefined or unknown. In [11] to deal with unknown and missing values, an incomplete context is defined as $\mathbb{K}_i = (O, A, \{+, ?, -\}, J)$ with an extension of KLEENE-logic is proposed. In our approach, we use a semi-lattice to represent missing and unknown values. We give the formal definition of a skeleton:

$$S = \{+ = \text{"existing, present"}, - = \text{"missing, absent"}, * = \text{"unknown,undefined"}\},$$

a skeleton is a labeled rooted tree $\mathcal{M}$ using the alphabet S i.e. each node $A_j \in \mathcal{A}$ of $\mathcal{M}$ is assigned a symbol from S. A map $\sigma : \mathcal{A} \to S$ defines a labeled rooted tree H_σ :

$$H_\sigma = (\mathcal{A}_\sigma, \mathcal{U}) \text{ with } \mathcal{A}_\sigma = \{(A_j, \sigma(A_j))_{j \in J}\}.$$

The skeleton nodes satisfy the following properties: the descendants of a missing (respectively unknown) node must be missing (respectively unknown). If a node is present, its children may be present, absent or unknown. Then, all the labeled rooted trees H_σ defined from a mapping $\sigma \in S^{\mathcal{A}}$ are not a valid representation of a skeleton object, it leads to:

Definition 1. *Let $B :< B_l >_{l \in L}$ be any structured attribute. The mapping $\sigma \in S^{\mathcal{A}}$ is said to be consistent, if it satisfies the following conditions:*
1) $\sigma(B) = - \Rightarrow \sigma(B_l) = -$ for $l \in L$, 2) $\sigma(B) = * \Rightarrow \sigma(B_l) = *$ for $l \in L$. The set of consistent maps is denoted $S_c^{\mathcal{A}}$.

We denote by $\mathcal{H}$ the set of skeletons related to consistent maps of $S_c^{\mathcal{A}}$.

Order Skeletons are defined from mapping $\sigma \in S_c^{\mathcal{A}}$. To order the skeleton space $\mathcal{H}$, it suffices to define an order on $S_c^{\mathcal{A}}$. The set $S = \{+, -, *\}$ is ordered as follows $* < +$ and $* < -$. In the context of information orderings, it means that $+$ and $-$ is more defined or precise than $*$. Let us notice that $+$ and $-$ are not comparable. Then $S^{\mathcal{A}}$ is pointwise ordered, for maps $s, s' \in S^{\mathcal{A}}$

$$s \leq s' \iff \forall A_j \in \mathcal{A}, s(A_j) \leq s(A_j)$$

The set $S^{\mathcal{A}}$ has **a minimum element** σ_* such as: $\forall A_j \in \mathcal{A}, s_*(A_j) = *$. The set $S_c^{\mathcal{A}} \subset S^{\mathcal{A}}$ inherits the pointwise order.

Semilattice We will define a semilattice structure on the skeleton set $\mathcal{H}$. The ordered set $(S = \{+, -, *\}, <)$ is a meet semilattice, because $* = + \wedge_S -$. It means that an undefined value is interpreted as a missing or existing node. Then $S^{\mathcal{A}}$ is also a meet semilattice, $\wedge_{S^{\mathcal{A}}}$ in $S^{\mathcal{A}}$ is defined from $\wedge_S$ as follows :

$$\forall A_j \in \mathcal{A}, s \wedge_{S^{\mathcal{A}}} s'(A_j) = s(A_j) \wedge_S s'(A_j).$$

Unfortunately, $S_c^{\mathcal{A}}$ is not a meet semilattice for $\wedge_{S^{\mathcal{A}}}$ because $S_c^{\mathcal{A}}$ is not stable under $\wedge_{S^{\mathcal{A}}}$. Consider $B =< B_1, B_2 >$ and $\sigma, \sigma' \in S_c^{\mathcal{A}}$ such as: $\sigma(B) = +, \sigma'(B) = -$; $\sigma(B_1) = +, \sigma'(B_1) = -$; $\sigma(B_2) = -, \sigma'(B_2) = -$. Then, we have: $\sigma(B) \wedge_{S^{\mathcal{A}}} \sigma'(B) = + \wedge_S - = *$; $\sigma(B_1) \wedge_{S^{\mathcal{A}}} \sigma'(B_1) = + \wedge_S - = *$; $\sigma(B_2) \wedge_{S^{\mathcal{A}}} \sigma'(B_2) = - \wedge_S - = -$

We see (Figure 2) that the value $-$ of the child B_2 of B is not equal to his father's value $*$, $\sigma \wedge_{S^{\mathcal{A}}} \sigma'$ is not consistent (we will say that the node B_2 is inconsistent for $\sigma \wedge_{S^{\mathcal{A}}} \sigma'$). Next proposition defines an operator $\wedge$ that associates

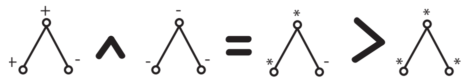


Fig. 2. Inf operator applied on simple skeletons.

a greatest lower bound in $S_c^{\mathcal{A}}$ to any $\sigma, \sigma' \in S_c^{\mathcal{A}}$.

Proposition 1. *Let $\sigma, \sigma' \in S_c^{\mathcal{A}}$. The set $S_c^{\mathcal{A}}$ is a meet semilattice such that:*

$$\sigma \wedge \sigma' = \bigvee([\sigma_*, \sigma \wedge_{S^{\mathcal{A}}} \sigma'] \cap S_c^{\mathcal{A}})$$

Proof. The set of all lower bounds of $\{\sigma, \sigma'\}$ is the interval $[\sigma_*, \sigma \wedge_{S^{\mathcal{A}}} \sigma']$ in $S^{\mathcal{A}}$. The set $[\sigma_*, \sigma \wedge_{S^{\mathcal{A}}} \sigma'] \cap S_c^{\mathcal{A}}$ is not empty because the minimum element $\sigma_* \in S_c^{\mathcal{A}}$. We are going to define the upper bound $\sigma \wedge \sigma'$ of $\{\sigma, \sigma'\}$ in $S_c^{\mathcal{A}}$. Let $B :< B_l >_{l \in L}$ be any structured attribute. Notice that if $\sigma \wedge_{S^{\mathcal{A}}} \sigma'(B) = -$, the node B does not lead to inconsistency. Hence, $\sigma \wedge_{S^{\mathcal{A}}} \sigma'(B) = - \Rightarrow \sigma(B) = \sigma'(B) = -$ as $\sigma, \sigma' \in S_c^{\mathcal{A}}$ for $l \in L, \sigma(B_l) = \sigma'(B_l) = -$ and $\sigma \wedge_{S^{\mathcal{A}}} \sigma'(B_l) = -$. Any inconsistency node B_l is such that $\sigma(B_l) \wedge_{S^{\mathcal{A}}} \sigma'(B_l) = -$ with $\sigma(B) \wedge_{S^{\mathcal{A}}} \sigma'(B) = *$. It means that the father B is present in one skeleton and missing in the other one and the child B_l is missing in the two skeletons (if B is indefinite in the two skeletons, all children will be indefinite because σ and σ' are consistent). In this case, we define $\sigma \wedge \sigma'(B_l) = *$. To sum up, we have $\sigma \wedge \sigma'(A_j) = \sigma \wedge_{S^{\mathcal{A}}} \sigma'(A_j)$ for all consistent nodes A_j, and $\sigma \wedge \sigma'(A_j) = *$ for all inconsistent nodes A_j. It is easy to see that $\sigma \wedge \sigma'$ is the greatest consistent lower bound of $\{\sigma, \sigma'\}$.

As $(S_c^{\mathcal{A}}; <, \wedge)$ is a meet semilattice, then the set of skeletons set $(\mathcal{H}; <, \wedge)$ is a meet semilattice, such that for $H_\sigma, H_{\sigma'} \in \mathcal{H}$:

$$H_\sigma \wedge H_{\sigma'} = H_{\sigma \wedge \sigma'}$$

The procedure that computes $\sigma \wedge \sigma'(A_j)$ is given below:

```
**Procedure** $\sigma \wedge \sigma'(A_j) = \bigwedge(\sigma(A_j), \sigma'(A_j))$

– If $A_j$ is a basic attribute then *return* $\sigma(A_j) \wedge_S \sigma'(A_j)$;
– elseif $A_j :< A_l >_{l \in L}$ is a structured attribute,
    • If $\sigma(A_j) = \sigma'(A_j) = +$ then { $\sigma \wedge \sigma'(A_j) = +$; For $l \in L, \sigma \wedge \sigma'(A_l) = \bigwedge(\sigma(A_l), \sigma'(A_l))$; } elseif $\sigma(A_j) = \sigma'(A_j) = -$ then *return* $-$ else *return* $*$;
```

The skeleton $H_\sigma \wedge H_{\sigma'}$ is built from the root down by applying, in breadth-first way, the procedure $\bigwedge(\sigma(A), \sigma(A))$. It stops when all common present structured attributes have been processed. Then, descendant of missing nodes must be labeled with $-$ and descendant of unknown nodes with $*$. The procedure $\bigwedge$, only on common present nodes, computes the greatest lower bound recursively this leads us to the definition of the **skeleton level**. Let $l(A_j)$ be the level number of the node A_j i.e. the length of the unique simple path from the root to A_j.

Definition 2. *The level $\nu(H_\sigma)$ of the skeleton H_σ is the largest level number of present nodes in H_σ : $\nu(H_\sigma) = max\{l(A_j) | \sigma(A_j) = +, A_j \in \mathcal{A})$*

4 Concepts Analysis on structured and multivalued data

In the Section 2, GC on semilattices has been introduced, and in previous sections a semilattice structure has been built on the skeleton set. Here, we apply these results to concepts determination for structured data.

Let denote by H_{σ_o} the skeleton of the object o, where $\sigma_o : \mathcal{A} \to S$. The mapping $d : O \to \mathcal{H}$ associates every element $o \in O$ with its description $d(o) = H_{\sigma_o}$. Consider the semilattice skeleton $(\mathcal{H}; <, \wedge)$ and $(\mathcal{P}(O); \subset, \cup)$. The pair $GC = (int, ext)$ of maps $ext : \mathcal{H} \longrightarrow \mathcal{P}(O)$ and $int : \mathcal{P}(O) \longrightarrow \mathcal{H}$, is a GC such as, for any $H_\sigma \in \mathcal{H}$:

$$ext(H_\sigma) = \{o \in O | H_\sigma \leq H_{\sigma_o}\}$$

and for $L \subset O$

$$int(L) = \bigwedge_{l \in L} H_{\sigma_l} = H_{\wedge_{l \in L} \sigma_l}.$$

The structure context is $\mathbb{K}_s = (O, \mathcal{H}, d)$, and the set of concepts induced by GC will be denoted by $\mathcal{C}$.

Let r be the height of the rooted tree $\mathcal{M} = (\mathcal{A}, \mathcal{U})$ i.e. the largest level number of a node, and let k be an integer such that $1 \leq k \leq r$. and

– $\mathcal{A}_k = \{A_j \in \mathcal{A} | l(A_j \leq k\}$ the set of attributes with a level less or equal to k,
– $\mathcal{M}_k = (\mathcal{A}_k, \mathcal{U}_k)$ the rooted tree such that the eight is k,
– $S_{\mathcal{C}}^{\mathcal{A}_k}$ the set of consistent mappings $\sigma^k : \mathcal{A}_k \to S$,
– $\mathcal{H}_k = \{H_{\sigma^k}\}$ the semilattice skeleton defined by $\mathcal{M}_k$,

- d_k the mapping $d_k : O \rightarrow \mathcal{H}_k$ such that $d_k(o) = H_{\sigma_o^k}$, the subtree of H_{σ_o} limited to nodes whose levels are less or equal to k.
- $GC_k = (int_k, ext_k)$ the GC, related to the context $\mathbb{K}_k = (O, \mathcal{H}_k, d_k)$, such that $ext_k(H_{\sigma^k}) = \{o \in O | H_{\sigma^k} \leq H_{\sigma_o^k}\}$ and $int_k(L) = \bigwedge_{l \in L} H_{\sigma_l^k} = H_{\wedge_{l \in L} \sigma_l^k}$,
- $\mathcal{C}_k$ the set of concepts induced by GC_k

The relationship between the set of concept $\mathcal{C}_k$ and $\mathcal{C}$ is precised by the following proposition

Proposition 2. *Let k be an integer $1 \leq k \leq r$, and let $c_k \in \mathcal{C}_k$ be a concept induced by GC_k. If the level $\nu(int_k(c_k))$ of the skeleton $int_k(c_k)$ is strictly less than k then it exists one concept $c \in \mathcal{C}$ induced by GC, such that its intension $int(c)_k$, limited to nodes whose levels are less or equal to k, is $int_k(c_k)$ and $ext(c) = ext_k(c_k)$. Conversely, if $c \in \mathcal{C}$ is a concept such that its level $\nu(int(c)) < r$ then, for any integer k such that $\nu(int(c)) \leq k \leq r$, $c_k = (int_k(c), ext(c))$ is a concept of $\mathcal{C}_k$.*

Proof. Let us note that for any skeleton H_σ, the projection of H_σ on $\mathcal{A}_k$ is H_{σ^k}. Let $c_k \in \mathcal{C}_k$, and denoted by $L = ext(c_k)$ and $H_{\sigma^k} = int(c_k) = \bigwedge_{l \in L} H_{\sigma_l^k}$. Consider that the level $\nu(H_{\sigma^k})$ is strictly less than k then nodes $A_j \in H_{\sigma^k}$, such that $l(A_j) = k$, is missing or unknown. There is an unique consistent skeleton $H_\sigma \in \mathcal{H}$, such that its projection on $\mathcal{A}_k$ is H_{σ^k}. H_σ is obtained by labeling the descendants of missing nodes, whose level is greater or equal to k, by $-$ and the descendants of unknown nodes of H_{σ^k}, whose level is greater or equal to k., by $*$. Consider that level $\nu(H_{\sigma^k}) < k$, and $H_{\sigma^k} = \bigwedge_{l \in L} H_{\sigma_l^k}$, this means that the objects of the extension L of c_k have not present nodes in common such that the level is greater than k, then $H_\sigma = \bigwedge_{l \in L} H_{\sigma_o}$, is the intension of L and $c = (H_\sigma, L)$ is a concept of $\mathcal{C}$ with the same extension than c_k.

Let $c \in \mathcal{C}$ whose intension is $int(c) = H_\sigma = \bigwedge_{o \in\ ext(c)} H_{\sigma_o}$, whose level $\nu(int(c)$ is strictly less than r. Let denote by H_{σ^k} the projection of H_σ at the level k. The level of H_σ is strictly less than r, then, for k such that $\nu(int(c)) \leq k \leq r$, we can state that $H_{\sigma^k} = \bigwedge_{o \in\ ext(c)} H_{\sigma_o^k}$. And c_k is a concept of $\mathcal{C}_k$ such that its intension is H_{σ^k} and its extension $ext(c_k) = ext(c)$.

This proposition gives a top down algorithm for structure concepts search. If c_k is a concept of $\mathcal{C}_k$, we can derive concepts c of $\mathcal{C}$ from c_k as follows

Procedure $\{c\} = DeriveConcepts(H_{\sigma^k} = int(c_k), ext(c_k))$

```
– Compute $A_k^+ = \{A_j \in \mathcal{A}_k | \sigma^k(A_j) = +, l(A_j) = k\}$;
– If $A_k^+ = \emptyset$ then return $c = c_k$,
– elseif $\{c_{k+1}\} = ConceptAnalysis(\mathbb{K}_{k+1} = (ext(c_k), \mathcal{H}_{k+1}, d_{k+1}))$;
– For each $c_{k+1}$ do $DeriveConcepts(int(c_{k+1}), ext(c_{k+1}))$;
```

The procedure $\{c_{k+1}\} = ConceptAnalysis(\mathbb{K}_{k+1} = (ext(c_k), \mathcal{M}_{k+1}, d_{k+1}))$ extracts concepts c_{k+1} from the extension of c_k. One can show that it may be implemented using a standard Formal Concept Analysis algorithm applied to the observations of c_k using only the attributes of A_k^+.

4.1 Semilattice on object values

In this section, we deal with the values of objects, we construct a meet semilattice structure on the values space of objects. Assume that is given a set of basic attributes names, $A_Q = \{(A_q\}_{q \in Q}$ and their corresponding domains $D_Q = \{D_q\}_{q \in Q}$. For any object o, a basic attribute is valued in D_q only if the attribute is present.

Denote by $\Gamma_q = D_q \cup \{\perp\} \cup \{*\}\}$ where $\perp$ is interpreted as "undefined" or "not applicable" values and will be used as the values for missing basic attributes, $*$ means that the value is unknown because the corresponding basic attribute is unknown. Denote by $\Gamma_Q = \sqcap_{q \in Q} \Gamma_q$. We assume that each set $D_q \in \mathcal{D}_Q$ is a meet semilattice according to the type of the basic attribute, it means that :

$$v_q, v'_q \in D_q \Rightarrow v_q \wedge v'_q \in D_q.$$

Consider $(\Gamma_q; <, \wedge, *)_{q \in Q}$ as the meet semilattice with $*$ as the minimum and the element $\perp$ is not comparable with $v_q \in D_q, v_q \wedge \perp = *$. Then $\Gamma_Q = \sqcap_{q \in Q} \Gamma_q$ is a meet semilattice as product of meet semilattice such as, for

$$v = (v_q)_{q \in Q}, v' = (v'_q)_{q \in Q} \in \Gamma_Q : v \wedge v' = (v_q \wedge v'_q)_{q \in Q} \in \Gamma_Q.$$

For example, for any categorical attribute (A_q, D_q), we will consider $D_q \cup \perp$ as an antichain, and the meet semilattice Γ_q has $*$ as minimum. If the type of a basic attribute is real interval, the domain is the set of values $u = [\underline{u}, \overline{u}]$ with $\underline{u}, \overline{u} \in \mathbb{R}$ such that $\underline{u} \leq \overline{u}$. The order relation chosen is the dual order of $\subset$, and the $\wedge$ operator is such that $u \wedge v = [\underline{u} \wedge \underline{v}, \overline{u} \vee \overline{v}]$.

The partial valuation function v_q related to the basic attribute A_q associates to each object o, a value $v_q \in \Gamma_q$ such as:

$$\sigma(A_q) = * \iff v_q = *;\ \sigma(A_q) = - \iff v_q = \perp;\ \sigma(A_q) = + \iff v_q \in D_q.$$

The valuation function $v : O \to \Gamma_Q$ is such as;

$$v(o) = (v_q(o))_{q \in Q} \text{ with } v_q(o) \in \Gamma_q$$

The value context is $\mathbb{K}_v = (O, \Gamma_Q, v)$.

Let $\delta : O \to \Gamma = \mathcal{H} \times \Gamma_Q$ be the mapping $d \times v$ which associates every object o with its skeleton $d(o) = H_{\sigma_o}$ and its values $v(o) = (v_q)_{q \in Q}$ taken on the basic attributes:

$$\delta(o) = d \times v(o) = (H_{\sigma_o}, v(o)) \in \Gamma = \mathcal{H} \times \Gamma_Q.$$

The conditions that the values must verify, lead us to

Definition 3. *Let $H_\sigma \in \mathcal{H}$ be a skeleton and let $v = (v_q)_{q \in Q} \in \Gamma_Q$, and let A_q be any basic attribute. (H_σ, v) is said to be consistent if σ and v satisfies the following conditions:*

$$\sigma(A_q) = * \iff v_q = *;\ \sigma(A_q) = - \iff v_q = \perp;\ \sigma(A_q) = + \iff v_q \in D_q.$$

In previous sections, we have shown how to provide the skeleton set $\mathcal{H}$ and Γ_Q with a meet semilattice structure. The description space $\Gamma = \mathcal{H} \times \Gamma_Q$ is a meet semilattice as a product of the meet semilattices $\mathcal{H}$ and Γ_Q. The greatest lower bound of the description of o and o' is written:

$$\delta(o) \wedge \delta(o') = (H_{\sigma_o \wedge \sigma_{o'}}, v(o) \wedge v(o'))$$

we shall ask the question : is this description consistent ? The next proposition shows that $\wedge$ preserves the consistency property

Proposition 3. *If (H_σ, v) and $(H_{\sigma'}, v')$ are consistent then $(H_{\sigma \wedge \sigma'}, v \wedge v')$ is consistent.*

Proof. Let $v = (v_q)_{q \in Q}$ and $v' = (v'_q)_{q \in Q}$ be values related to consistent descriptions (H_σ, v) and $(H_{\sigma'}, v')$. Assume A_q a basic attribute such that $\sigma \wedge \sigma'(A_q) = *$. Then, the first possibility is $\sigma(A_q) = \sigma'(A_q) = * \iff v_q = v'_q = *$, because the descriptions are consistent, then we have $v_q \wedge v'_q = *$. Or A_q is missing in one skeleton and present in the other one. Let us suppose that $\sigma(A_q) = + \iff v_q \in D_q$, and $\sigma'(A_q) = - \iff v'_q = \bot$. We always have $v_q \wedge v'_q = v_q \wedge \bot = *$, and conversely, if A_q is a basic attribute such that $\sigma \wedge \sigma'(A_q) = -$, then $\sigma(A_q) = \sigma'(A_q) = -$, and $v_q = v'_q = \bot$,then we have $v_q \wedge v'_q = \bot$. $(H_{\sigma \wedge \sigma'}, v \wedge v')$ is consistent.

4.2 Concepts

The goal of the previous sections has been to define a complex context $\mathbb{K} = (O, \Gamma, \delta)$ for structured, and multi-valued and incomplete data.

Let $\Gamma(O)$ be the meet semilattice generated by the descriptions of the objects $\Gamma(O) = \{\wedge_{l \in L} \delta(l) | L \subset O\} = \{\wedge_{l \in L} (H_{\sigma_l}, v(l)) | L \subset O\}$. Consider the semilattice $(\Gamma(O), <, \wedge)$ and $(\mathcal{P}(O), \subset, \cup)$. The pair $GC = (int, ext)$ of maps
$ext : \Gamma(O) \longrightarrow \mathcal{P}(O)$ and $int : \mathcal{P}(O) \longrightarrow \Gamma(O)$, is a GC such as, for $L \in \mathcal{P}(O)$:

$$int(L) = \bigwedge_{l \in L} (H_{\sigma_l}, v(l)) = (H_{\wedge_{l \in L} \sigma_l}, \wedge_{l \in L} v(l))$$

which is a consistent description according to the previous proposition, and for any $(H_\sigma, \nu) \in \Gamma(O)$:

$$ext((H_\sigma, \nu)) = \{o \in O | (H_\sigma, \nu) \leq (H_{\sigma_o}, v(o))\} = \{o \in O | H_\sigma \leq H_{\sigma_o}, \nu \leq v(o)\}.$$

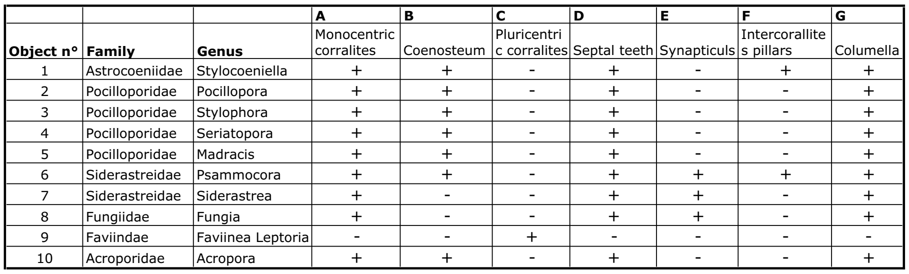
The set of concepts induced by GC is denoted by $\mathbb{C}$. The relationship between skeleton concepts of $\mathcal{C}$ and complex concepts of $\mathbb{C}$ is made precise below:

Proposition 4. *Let L be a set of objects, $\sigma_L = \wedge_{l \in L} \sigma_l$ and $v_L = \wedge_{l \in L} v(l)$. If $\gamma = (H_{\sigma_L}, L) \in \mathcal{C}$ is a skeleton concept, and if we have for any basic attribute $Aq, \sigma_L(A_q) \neq +$ then $\Upsilon = ((H_{\sigma_L}, v_L), L)$ is a complex concept of $\mathbb{C}$.*

Proof. The proof is easy as we can notice, that all basic values are missing or unknown if the corresponding basic attributes are missing or unknown.

5 Application: concepts extraction from coral base

A straight-forward application is conducted on coral description base. The concepts lattice is extracted from information about the structure of objects. One concept is exhibited with its resulting multi-values properties. We consider for this application a subset of 10 descriptions extracted from the coral genera base (16 families, 58 genera, 185 species). The whole knowledge base is actually made of 10 models corresponding to the 10 main coral families present in the south-west of the Indian ocean and about two thousands descriptions (see Figure 1). In order to use classical FCA methods, each structured attribute B is coded by two binaries attributes $B+$ and $B-$ to express the presence or absence of a component. For a given object o, an unknown state is represented by $B+(o) = B-(o) = 0$. Following table shows the resulted context:



			A	B	C	D	E	F	G
Object n°	**Family**	**Genus**	Monocentric corralites	Coenosteum	Pluricentric corralites	Septal teeth	Synapticuls	Intercorallites pillars	Columella
1	Astrocoeniidae	Stylocoeniella	+	+	-	+	-	+	+
2	Pocilloporidae	Pocillopora	+	+	-	+	-	-	+
3	Pocilloporidae	Stylophora	+	+	-	+	-	-	+
4	Pocilloporidae	Seriatopora	+	+	-	+	-	-	+
5	Pocilloporidae	Madracis	+	+	-	+	-	-	+
6	Siderastreidae	Psammocora	+	+	-	+	+	+	+
7	Siderastreidae	Siderastrea	+	-	-	+	+	-	+
8	Fungiidae	Fungia	+	-	-	+	+	-	+
9	Faviindae	Faviinea Leptoria	-	-	+	-	-	-	-
10	Acroporidae	Acropora	+	+	-	+	-	-	+

Fig. 3. An example of corals data set

We used the Galicia platform [12] and the Bordat algorithm [2] with the classical inf operator $\wedge_{\mathcal{SA}}$ to build the concepts semilattice (Figure 4) on the previous context. Each concept is presented with its intension, extension and the associated skeleton. We verify that C2, C3, C8 and C9 are inconsistent concepts: for them, the structured attribute A is undefined whereas at least one of its subpart is defined. At this stage, a concept regroups objects having similar skeletons. The interest to use the consistent inf operator $\wedge$ (see proposition 1) is that inconsistent concepts are not computed. The concept C11 groups the different Pocilloporidae family's genera and the quite near genus Acropora of the Acroporidae family. From the expert's point of view, this analysis is meaningful to organize taxonomies, according to M. Pichon, an nternational coral expert. From the extension of skeleton concepts, further analysis such as Concept Analysis on multivalued contexts or clustering methods can be performed. For example, Figure 5 gives the intension of the concept C11 computed, using IKBS system, from the complete objects description of the extension of C11.

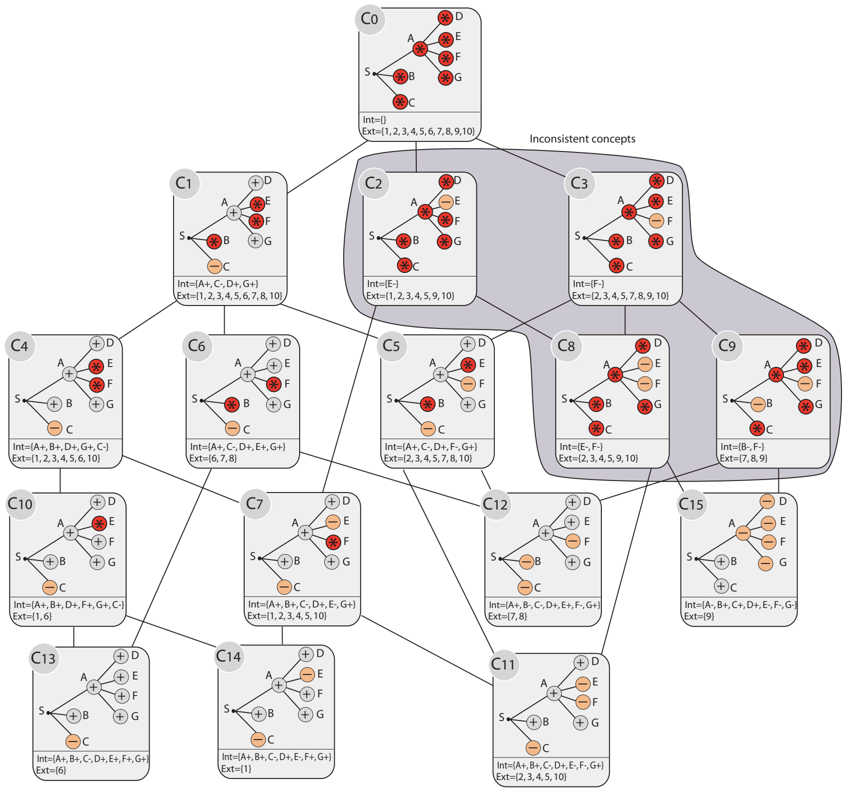
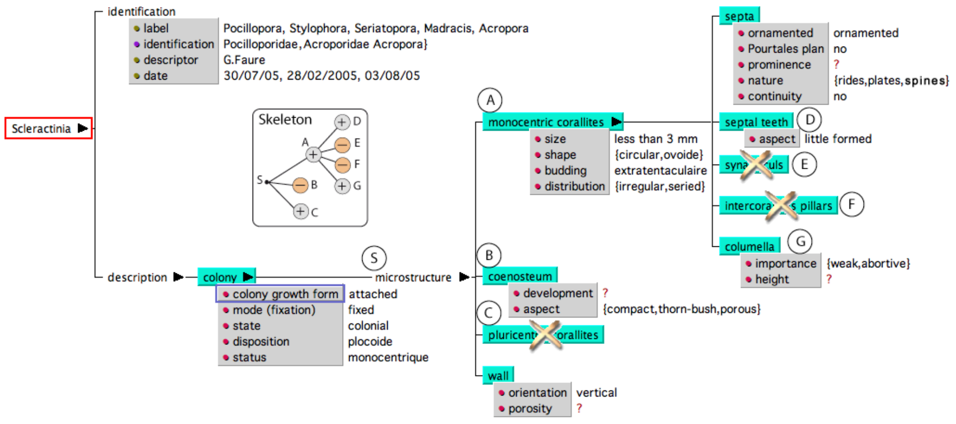

Inconsistent concepts
C0 Int={} Ext={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
C1 Int={A+, C-, D+, G+} Ext={1, 2, 3, 4, 5, 6, 7, 8, 10}
C2 Int={E-} Ext={1, 2, 3, 4, 5, 9, 10}
C3 Int={F-} Ext={2, 3, 4, 5, 7, 8, 9, 10}
C4 Int={A+, B+, D+, G+, C-} Ext={1, 2, 3, 4, 5, 6, 10}
C6 Int={A+, C-, D+, E+, G+} Ext={6, 7, 8}
C5 Int={A+, C-, D+, F-, G+} Ext={2, 3, 4, 5, 7, 8, 10}
C8 Int={E-, F-} Ext={2, 3, 4, 5, 9, 10}
C9 Int={B-, F-} Ext={7, 8, 9}
C10 Int={A+, B+, D+, F+, G+, C-} Ext={1, 6}
C7 Int={A+, B+, C-, D+, E-, G+} Ext={1, 2, 3, 4, 5, 10}
C12 Int={A+, B-, C-, D+, E+, F-, G+} Ext={7, 8}
C15 Int={A-, B+, C+, D+, E-, F-, G-} Ext={9}
C13 Int={A+, B+, C-, D+, E+, F+, G+} Ext={6}
C14 Int={A+, B+, C-, D+, E-, F+, G+} Ext={1}
C11 Int={A+, B+, C-, D+, E-, F-, G+} Ext={2, 3, 4, 5, 10}

Fig. 4. Concepts semilattice build upon structured objects with $\wedge_{S\mathcal{A}}$.



identification

label	Pocillopora, Stylophora, Seriatopora, Madracis, Acropora
identification	Pocilloporidae, Acroporidae Acropora}
descriptor	G.Faure
date	30/07/05, 28/02/2005, 03/08/05

description ▶ colony ▶ microstructure ▶

colony growth form	attached
mode (fixation)	fixed
state	colonial
disposition	plocoide
status	monocentrique

monocentric corallites ▶

size	less than 3 mm
shape	{circular,ovoide}
budding	extratentaculaire
distribution	{irregular,seried}

septa

ornamented	ornamented
Pourtales plan	no
prominence	?
nature	{rides,plates,spines}
continuity	no

septal teeth

aspect	little formed

columella

importance	{weak,abortive}
height	?

coenosteum

development	?
aspect	{compact,thorn-bush,porous}

wall

orientation	vertical
porosity	?

Fig. 5. Intension of the concept C11

6 Conclusion

In this paper, we presented an approach which allows Concept Analysis to deal with structured, multivalued and incomplete data. This kind of analysis is useful to extract knowledge from observations in Life Sciences and to help experts in the Knowledge Bases building process. However, the number of consistent concepts generated may be huge due to model's complexity. We are exploring strategies to reduce the concepts research space by using datamining methods such as clustering or suitable distances on structured and multi-valued objects.

References

[1] M. Barbut and B. Monjardet. *Ordre et classification*. Hachette, Paris, 1970.

[2] J.P. Bordat. Calcul pratique du treillis de galois d'une correspondance. In *Mathématiques et Sciences humaines, 96*, pages 31–47, 1986.

[3] P. Cimiano, A. Hotho, G. Stumme, and J. Tane. Conceptual knowledge processing wih formal concept analysis and ontologies. In P.W. Eklund, editor, *Concept Lattices, Second International Conference on Formal Concept Analysis (ICFCA 2004), Sydney, Australia*, LNCS 2961, pages 189–207, 2004.

[4] N. Conruyt and D. Grosser. Knowledge engineering in environmental sciences with ikbs: Application to systematics of corals of the mascarene archipelago. *AI Communication*, 16(4):267–278, 2003.

[5] Grosser D. *Construction itérative de bases de connaissances descriptives et classificatoires avec la plate-forme à objets IKBS : application à la systématique des coraux des Mascareignes*. Thèse de doctorat, Université de la Réunion, 2002.

[6] R. Emilion E. Diday. Maximal and stochastic galois lattices. *Discrete Applied Mathematics*, 27(2):271–284, 2003.

[7] B. Ganter and R. Wille. *Formal concept analysis, Mathematical foundations*. Springer Verlag, Berlin, 1999.

[8] Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. *Lecture Notes in Computer Science*, 2120:129–142, 2001.

[9] Le Renard J. and Conruyt N. On the representation of observational data used for classification and identification of natural objects. *LNAI IFCS'93*, pages 308–315, 1994.

[10] H.Ralambondrainy J. Diatta. The conceptual weak hierarchy associated with a dissimilarity measure. *Mathematical Social Sciences*, pages 301–319, 2002.

[11] Burmeister P. and Holzer R. Treating incomplete knowledge in formal concept analysis. *LNAI 3626*, pages 114–126, 2005.

[12] C. Roume P. Valtchev, D. Grosser and M. Rouane Hacene. Galicia, an open platform for lattices. In *Proceedings of the 11th International Conference on Conceptual Structures (ICCS'03)*, pages 241–254. Shaker Verlag, 2003.

[13] P. Valtchev, R. Missaoui, and R. Godin. Formal concept analysis for knowledge discovery and data mining: The new challenges. In P.W. Eklund, editor, *Concept Lattices, Second International Conference on Formal Concept Analysis (ICFCA 2004), Sydney, Australia*, LNCS 2961, pages 352–371. Springer, 2004.

[14] R. Wille. Methods of conceptual knowledge processing. In R. Missaoui and J. Schmid, editors, *International Conference on Formal Concept Analysis (ICFCA 2006), Dresden, Germany*, LNCS 3874, pages 1–29. Springer, 2006.

Using lattices for reconstructing stemma

Marc Le Pouliquen

1. Département LUSSI TAMCIC, UMR CNRS 2872
ENST Bretagne, BP 832, 29285 Brest Cedex
2. Université de Bretagne Occidentale IUP GMP
6 av. Le Gorgeu - CS93837 29238 Brest Cedex 3
{marc.lepouliquen@enst-bretagne.fr}

Abstract. The product of textual criticism is an edited text that the editor believes comes as close as possible to a lost original manuscript called the archetype. Usually, the editor compares different manuscripts of a single text, and represents it as an inverted tree showing all the steps in the transmission of a specific text, reconstructed by establishing relationships with other manuscripts. This tree is called the "stemma codicum"(cf. [7]). Because of the graphic proximity of the stemma with a semi-lattice, we propose to use two lattice construction techniques in order to reconstitute the filiation tree of manuscripts. First, we try the traditional methods to build the lattice of a binary relation (cf. [13]). Then a more specific solution to the problem is proposed. These techniques are finally tested on a real corpus of manuscripts by Rimbaud, "Les Effarés" (cf. [17]).

1 Introduction

In this paper, we use lattices as a pattern for the construction of the family tree of manuscripts within the framework of the critical edition.As far as possible, the editor must try to reconstitute , the original manuscript[1] as the author wrote it, starting from the various preserved manuscripts. The corpus is made up with many manuscripts which are copied from each other. To do so, it appears interesting to draw up a family tree of these manuscripts called the "stemma codicum".

As can be seen on Figure 1, the stemma is a kind of graph or a tree. We will extract our stemma from a lattice by pruning vertices and edges. The lattice is built starting from a binary relation between the manuscripts and their differences. This information is contained in the collation table[2]. Two methods are proposed to carry out the lattice pruning:

- An expert (in this case an editor) orders the most relevant concepts (in this case the "differences" between the manuscripts) according to his judgment.
- An algorithm helps the expert by removing the lattice vertices which have not enough "difference" on each level. After many iterations, the lattice becomes a tree or a graph representing a stemma.

This paper is organized as follows: In section 2, we persent philological methods for the establishment of stemma. In section 3, we describe visualization techniques for the building of the stemma which are tested in section 4 on a real corpus of poems.

[1] The **original manuscript** or archetype is the most recent common ancestor of all extant manuscripts in an textual tradition.
[2] **collation** is the comparison between a manuscript and the other manuscripts from the corpus for the sake of producing a list of the differences

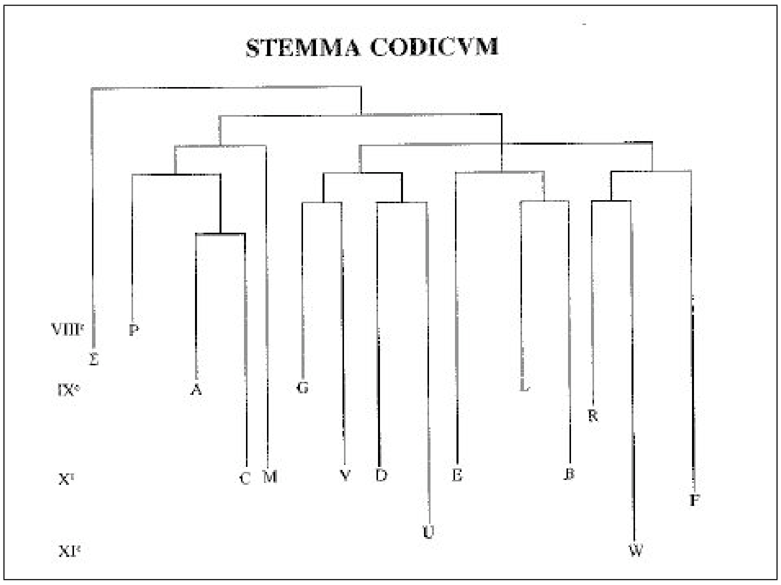


Fig. 1. Stemma codicum established by Reydellet in connection with Poems of Venance Fortunat [6]

2 Some philological methods to establish the stemma codicum

A text which was copied several times constitutes a “textual tradition” and all the specimens that have reached us are called the “witnesses”. Usually, the editor compares different witnesses of a single text, and makes a selection of variants (“readings”) taken from many sources to restore the original manuscript. The editors use a stemma codicum to evaluate readings, and vice versa.

Historically, several methods have been developed in order to try to visualize the genealogical relations between manuscripts. One of these methods, formalized by Lachmann[7] is now called the common error method. If an error is introduced into a manuscript, it is likely that the “descendant” of that text will show the same common error. So, a family of manuscripts is composed of the texts that have the same reading. Although this method has been largely criticized, both this method and its improvements have become indispensable to describe the history of the text

Another historical method is the method of Don Quentin[9]. He came up with the idea of reconstituting the sequence of the manuscripts by means of a three by three comparison. In fact, he assembled small chains of three manuscripts, one being between the other two and assembled these small chains in order to infer the complete tree.

After some counting, we notice that the number of different diagrams is exponentially dependent on the number of manuscripts. It is therefore impossible to consider all the stemmas, their construction and their comparison. This can explain why editors had difficulties formalizing both stemmas and their use. Thanks to the new visualization possibilities offered by lattices, the “stemmatization”methods can be modified and adapted to model the history of the text.

3 Lattice construction starting from binary relations

3.1 Algorithms and software

The last few years, Galois lattices of binary relations have been fields of important research in formal concept analysis (FCA) in particular for the visualization of many problems. FCA was basically inspired by the work of Birkhoff[3], Galois lattices were described by Barbut and Monjardet[1] and the whole approach was formalized by Ganter and Wille[11]. The lattice construction starting from binary relations and their visualization by the intermediary of the Hasse diagram allows greater comprehension of the binary table. Many algorithms have been developed for this construction:

- Those which build lattices in an incremental way i.e. they can update the lattice concept when a new object is added without re-computing the whole lattice
  - Godin[12]*
  - Carpineto and Romano[4]*
  - Norris[15]
- The other algorithms have to know the whole binary table before computing the lattice
  - Chein[5]
  - Ganter[10]
  - Bordat[2]*
  - Nourine and Raynaud[16]*

A detailed description of these algorithms and a comparison of lattice algorithms has been done in Guénoche[13] and Kuznetzov[14]. The methods which interest us are those followed by a * symbol because they can generate the Hasse diagram of Galois lattices.

In the experimentation, we need to visualize the lattices in a Hasse diagram to analyze them. With this purpose we opt for using two software for lattices representation:

- ConExp (Concept Explorer of Yevtushenko[19]) combines the creation and the visualization of the binary table in a simple tool. A view of the ConExp interface is in Figure 5. The diagrams can be exported to the JPEG or GIF format. With ConExp, it is possible to carry out many operations of Ganter and Wille[11].
- Galicia is the interactive lattice construction tool of Valtchev et al.[18]. Simple and valued contexts can be analyzed. The binary relations and the objects can also be described and stored. The lattices can be saved in JPEG, SVG or PDF formats (cf. Fig. 4).

3.2 Simple example

For the example, let us reduce the manuscripts to three sentences. Let there be the three following sentences, which correspond to the same sentence of manuscripts that were copied one from the other.

Mns1 =“Here is a sentence invented for the example”
Mns2 = “This is a sentence invented for the example”
Mns3 = “Here is a sentence built for the example”

There are two variant places here: (*Here/This*) and (*invented/built*), corresponding to four variants, as summarized in the collation table 1:

N° of variant place	Variants	Manuscripts	Variants	Manuscripts
1	Here	Mns1,Mns3	This	Mns2
2	invented	Mns1,Mns2	built	Mns3

Table 1. Collation table of our three manuscripts

Manuscripts/variants	Var1	Var1b	Var2	Var2b
Manuscript1	×		×	
Manuscript2		×	×	
Manuscript3	×			×

Table 2. Binary table

To obtain a binary table, we assign to the three manuscripts a boolean value according to the presence or absence of each variant (cf. Table 2).

From this table, we assume that the history of the text is summarized in the following way:

- Either (a) manuscript 2 is the nearest manuscript to the original. So here, manuscript 1 is copied from manuscript 2. The scribe modifies *This* in *Here*. Manuscript 3 is copied from manuscript 1, and another scribe modifies *invented* in *built*. On the other hand, if the first scribe modifies *Here* in *This*, there is little chance that the following scribe could find again *Here* by modifying *This*. It is not a credible assumption and one can say that the manuscript 1 is an intermediate between the manuscripts 2 and 3.
- Or (b) manuscript 1 is the nearest manuscript to the original and manuscripts 2 and 3 are copied from it
- Or (c) manuscript 3 is the nearest manuscript to the original, then manuscript 1 is copied from it and manuscript 2 copied from manuscript 1.

According to the information contained in the sentences, it is not possible to choose between these three stemmas in Figure 2 without the help of external information as datation or codicologic studies[3].

$$\begin{tikzcd}
Mns2 \arrow[d] \\
Mns1 \arrow[d] \\
Mns3
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
 & Mns1 \arrow[dl] \arrow[dr] & \\
Mns2 & & Mns3
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
Mns3 \arrow[d] \\
Mns1 \arrow[d] \\
Mns2
\end{tikzcd}$$

(c)

Fig. 2. Three possible stemmae

We now associate the binary table (cf. Table. 2) to the lattice (cf. Fig. 3) obtained by the previous algorithms (cf. 3.1). We note that we obtain a perfect representation of the manuscripts and their variants; indeed the lattice shows that manuscripts 1

[3] **Codicologic information** are for example: the color of the ink, the order of the page who can be modified over the time...

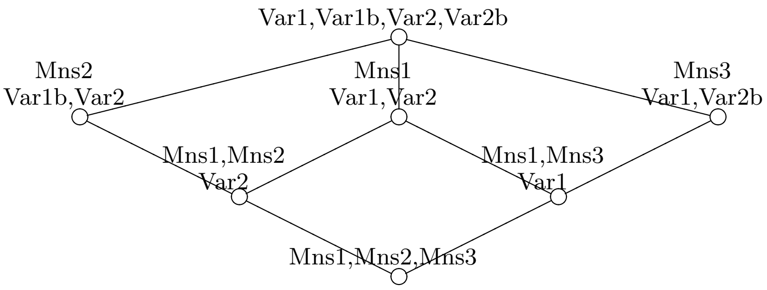


Fig. 3. Lattice

and 2 have Var2 (*invented*) as their common variant. To obtain the stemma starting from the Hasse diagram, two methods are proposed:

- the editor must remove the less significant variants until the lattice becomes a tree. In our example, if the editor assumes that Var1 is more judicious than Var1b, he obtains Figure 4 and this choice is called "emendation"[4]. After if the editor prefers Var2b to Var2, we obtain the Hasse diagram of Figure 5 which corresponds to the preceding stemma (c).

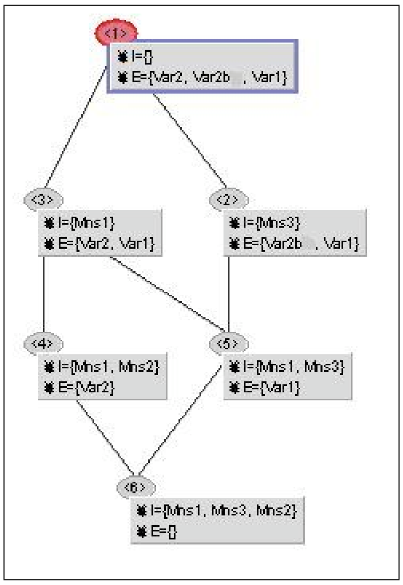


Fig. 4. Lattice using Galicia by removing Var1b

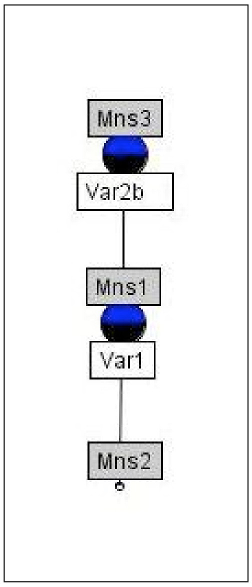


Fig. 5. Lattice using ConExp by removing Var1b and Var2

- An algorithm helps the editor by removing the lattice vertices witch have not enough relevance.

Algorithm 1

1. *We delete the vertice which have the least variants by level*

[4] **emendation**, a correction made to a text in the belief that the author's original wording has been wrongly altered

2. *We delete the variant of the same variant place which is contained in the previous deleted vertice.*
3. *We reiterate the process until the editor decides to stop it or as long as the graph remains connected.*
4. *REM: If the vertices have the same number of variants, we keep the variant of the same variant place which is contained in most manuscripts to begin the iteration.*

As in our example, there are only 2 variant places, so the algorithm keeps only Var1 and Var2. Actually, in the first variant place, Var1 is contained in manuscripts 1 and 3 whereas Var1bis is contained in manuscript 2 only, so the algorithm keeps Var1. Finally, we obtain the stemma (b) of the Figure 2. In this algorithm, the editor must constantly be able to impose his expert point of view on the interface which will be taken into account when realizing the stemma.

4 Application to a real corpus

We test this method on a real corpus of Rimbaud poems, "Les Effarés" or "Petit Pauvre" put together by Steve Murphy (cf. [17]). Here we consider five versions of this poem:

- GM reproduction of *Gentleman's Magazine* (1878)
- L Lutèce printed book (1883)
- JA Manuscripts' autograph[5] of Jean Aicard (1871)
- PD Manuscripts' autograph in Demeny's collection (1871)
- PV Copy of P. Verlaine (1872)

After collation (cf. Table 3) we use the same method described above, and investigate 14 different variants.

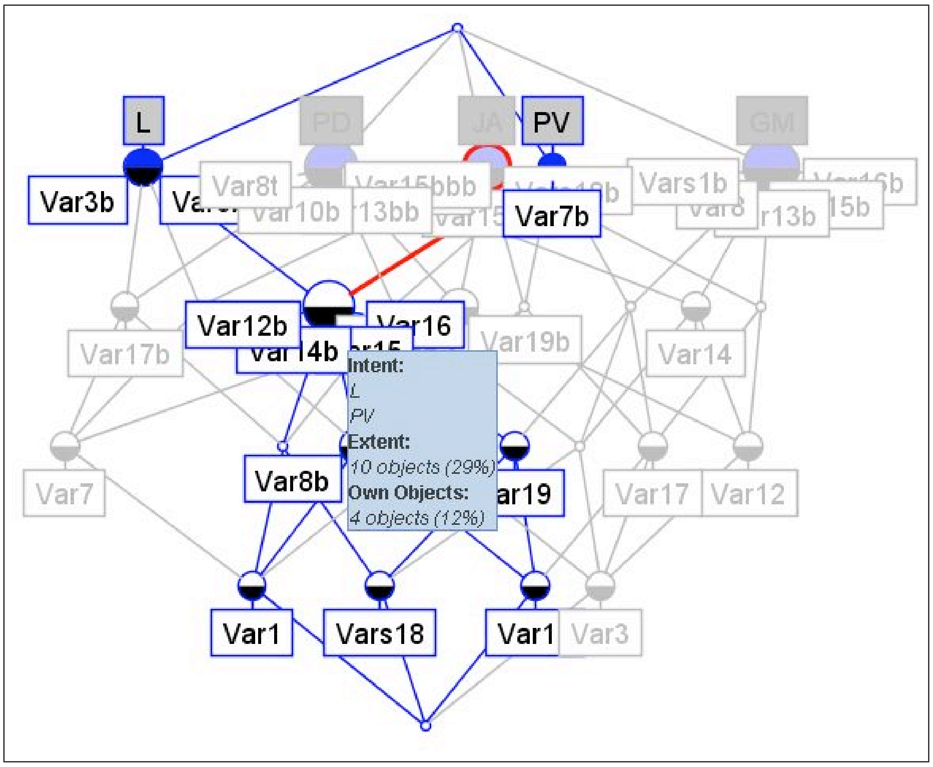
Variant place	Line	GM	L	JA	PD	PV
1	titre	Petits Pauvres	Les Effarés	Les Effarés	Les Effarés	Les Effarés
2	5	dos	culs	culs	culs	culs
3	7	cinq	Les	cinq	cinq	cinq
4	9	beaux	lourd	lourd	lourd	lourd
5	11-17		Ils voient	Ils voient	I ls voient	Ils voient
6	16		gros	gras	gras	gras
7	17		Chante	Chante	Chante	Grogne
8	23-25	boulanger	médianoche	médianoche	minuit sonne	médianoche
...	...	...	...	...	...	...

Table 3. Collation table of poems

With ConExp, we achieve a lattice that we will use to find the stemma. Initially, the visualization shows that common variants of L and PV (cf. Fig. 6) represent 10 variants out of 14. A high score of common variants means a close relation between these two manuscripts, for L is assumed to be a reconstruction from memory by Verlaine.

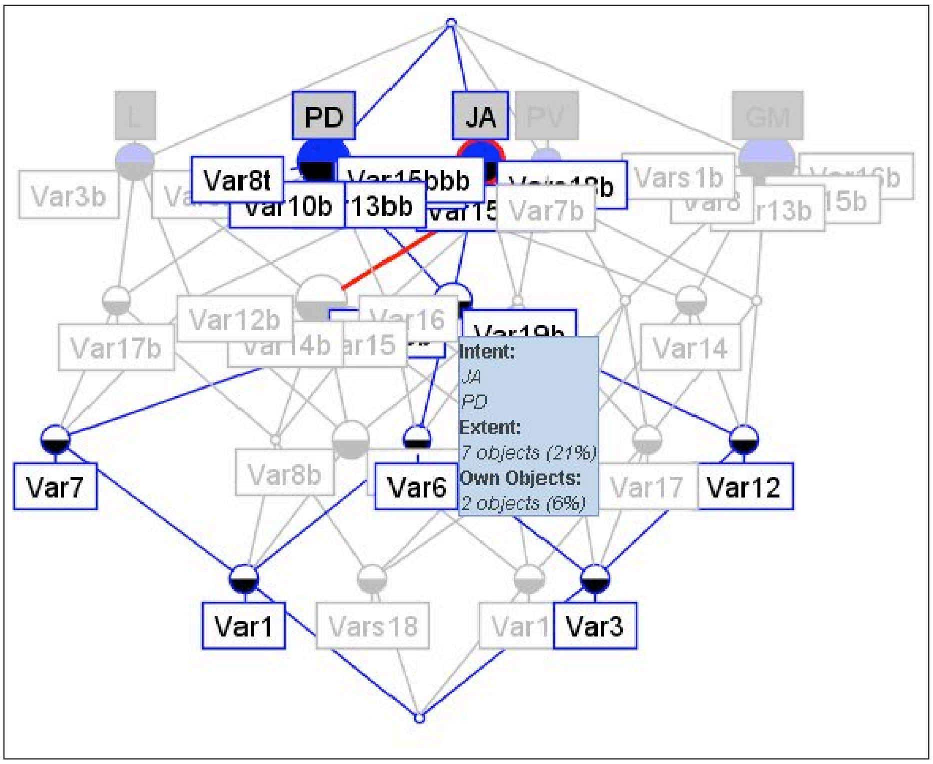
The second diagram (cf. Fig. 7) shows us that the two manuscripts JA and PD have the second hightest score of common variants. They are manuscripts that are the oldest and closest by date, which may explain their proximity.

[5] An **autograph** is a document written entirely in the handwriting of its author



Intent:
L
PV
Extent:
10 objects (29%)
Own Objects:
4 objects (12%)

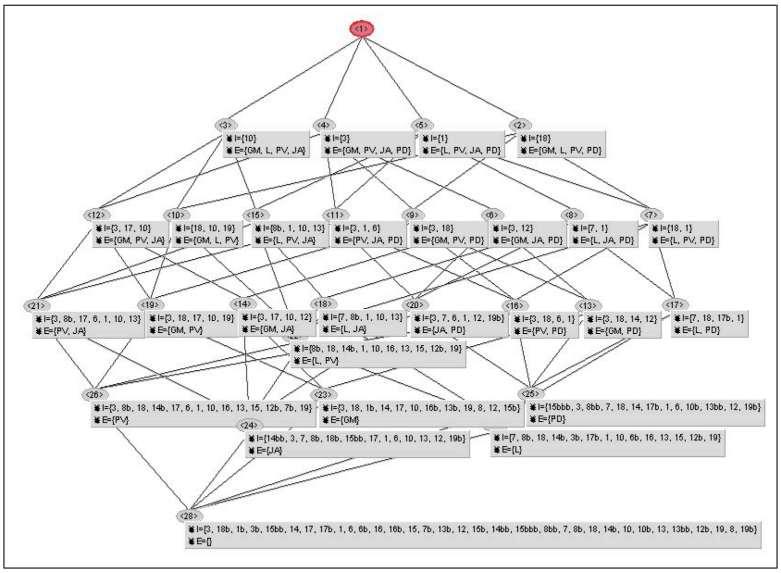
Fig. 6. Hasse diagram of Les Effarés. Relation between manuscripts PV and L



Intent:
JA
PD
Extent:
7 objects (21%)
Own Objects:
2 objects (6%)

Fig. 7. Hasse diagram of Les Effarés. Relation between manuscripts JA and PD

To attempt to extract a family tree (stemma) from the Hasse diagram (cf. Fig. 8), we cannot extract the most significant variants, because they are almost all legitimate and undoubtedly by the author himself. We use the algorithm which involves in removing the least significant vertices i.e. those which have fewer variants on each level. Then we obtain Figure 9. If we continue the extraction as long as the "lattice" remains connected, we finish in removing vertice 4 and the dotted line edges. In this corpus, the stemma probably represents the proximity of the poems rather than a hypothetical filiation.



<1>
<3> I={10} E={GM, L, PV, JA}
<4> I={3} E={GM, PV, JA, PD}
<5> I={1} E={L, PV, JA, PD}
<2> I={18} E={GM, L, PV, PD}
<12> I={3, 17, 10} E={GM, PV, JA}
<10> I={18, 10, 19} E={GM, L, PV}
<15> I={8b, 1, 10, 13} E={L, PV, JA}
<11> I={3, 1, 6} E={PV, JA, PD}
<9> I={3, 18} E={GM, PV, PD}
<6> I={3, 12} E={GM, JA, PD}
<8> I={7, 1} E={L, JA, PD}
<7> I={18, 1} E={L, PV, PD}
<21> I={3, 8b, 17, 6, 1, 10, 13} E={PV, JA}
<19> I={3, 18, 17, 10, 19} E={GM, PV}
<14> I={3, 17, 10, 12} E={GM, JA}
<18> I={7, 8b, 1, 10, 13} E={L, JA}
<20> I={3, 7, 6, 1, 12, 19b} E={JA, PD}
<16> I={3, 18, 6, 1} E={PV, PD}
<13> I={3, 18, 14, 12} E={GM, PD}
<17> I={7, 18, 17b, 1} E={L, PD}
I={8b, 18, 14b, 1, 10, 16, 13, 15, 12b, 19} E={L, PV}
<26> I={3, 8b, 18, 14b, 17, 6, 1, 10, 16, 13, 15, 12b, 7b, 19} E={PV}
<23> I={3, 18, 1b, 14, 17, 10, 16b, 13b, 19, 8, 12, 15b} E={GM}
<25> I={15bbb, 3, 8bb, 7, 18, 14, 17b, 1, 6, 10b, 13bb, 12, 19b} E={PD}
<24> I={14bb, 3, 7, 8b, 18b, 15bb, 17, 1, 6, 10, 13, 12, 19b} E={JA}
I={7, 8b, 18, 14b, 3b, 17b, 1, 10, 6b, 16, 13, 15, 12b, 19} E={L}
<28> I={3, 18b, 1b, 3b, 15bb, 14, 17, 17b, 1, 6, 6b, 16, 16b, 15, 7b, 13b, 12, 15b, 14bb, 15bbb, 8bb, 7, 8b, 18, 14b, 10, 10b, 13, 13bb, 12b, 19, 8, 19b} E={}

Fig. 8. Hasse diagram of Les Effarés using Galicia

5 Prospects and Conclusion

In future work, we therefore plan to write a program designed to be used as a stemma construction aid for the textual scholar. In all cases, the editor may interact with the program to improve the results using human insight. This interaction is necessary if we want to persuade editors of the usefulness and the interest of the system. The use of lattices is necessary to visualize the relations between manuscripts and their variants and providing the editor with the required interactions.

However, many tasks remain:

- Sometimes the corpus contains more than one hundred manuscripts and one thousand variants. Under these conditions, how can we optimize the visualization?
- Many statistical and probabilistic aspects must also be considered during the automatic lattice pruning.
- Methods based on phylogenetic trees are already used for drawing stemmas (cf. [8]); how can we combine these two methodologies ?

References

1. Barbut M. and Monjardet B. : Ordre et classification : Algèbre et combinatoire, *Collection Hachette Université*, (1970)

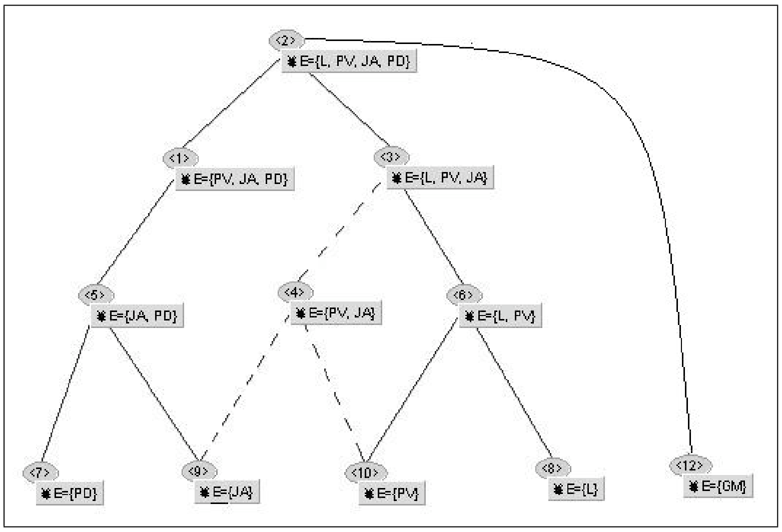


Fig. 9. Stemma of Les Effarés

2. Bordat J.P. : Calcul pratique du treillis de Galois dune correspondance, *Math. Sci. Humaines* No 96 (1986), 31–47
3. Birkhoff G. : Lattice Theory, *American Mathematical Society Publications*, (1940)
4. Carpineto C. and Romano G.: A lattice conceptual clustering system and its application to browsing retrieval, *Machine Learning* No 24 (1996), 95–122
5. Chein M. : Algorithme de recherche des sous-matrices premières d'une matrice, *Bull. Math. Soc. Sci. Math. R.S. Roumanie* No 13 (1969), 21–25
6. Fortunat V. and Reydellet M. : Poèmes Livres I-IV, *Belles lettres*, (1994)
7. Lachmann, K.K.F.W. : Kleinere Schriften, *Berlin*, (1876)
8. Le Pouliquen M., Barthélemy J.P. and Bertrand P. : Filiation de manuscrits sanskrits et arbres phylogénétiques, *soumise en vue d'une parution dans un numéro spécial de la revue RNTI (SFC06)* , (2006)
9. Quentin, H. : Essais de critique textuelle, *Picard*, (1926)
10. Ganter B. : Two Basic Algorithms in Concept Analysis, *Technische Hochschule Darmstadt*, Preprint No. 831 (1984)
11. Ganter B. and Wille R. : Formal Concept Analysis, Mathematical foundations, *Springer-Verlag*, (1999)
12. Godin R., Missaoui R., and Alaoui H.: Incremental concept formation algorithms based on galois lattices, *Computation Intelligence*, No. 11(2) (1995), 246–267
13. Guenoche A. : Construction du treillis de Galois d'une relation binaire, *Math. Inform. Sci. Humaines*, No. 109 (1990), 41–53
14. Kuznetsov S.O. and Obiedkov S.A. : Comparing performance of algorithms for generating concept lattices, *Exp. Theoret. Artificial Intelligence*, No 14 23 (2002), 189–216
15. Norris E.M. : An Algorithm for Computing the Maximal Rectangles in a Binary Relation, *Revue Roumaine de Mathématiques Pures et Appliquées*,No. 23(2) (1978), 243–250
16. Nourine L. and Raynaud O. : A fast algorithm for building lattice, *Information Processing Letters*, (1999), 199–204
17. Rimbaud, A. : Poésies, édition critique avec introduction et notes de Steve Murphy, *Honoré Champion*, (2000)
18. Valtchev P., Gosser D., Roume C. and Hacene M.: Galicia: an open platform for lattices, *In Aldo de Moor, Wilfried Lex, and Bernhard Ganter, editors, Contributions to the 11th Conference on Conceptual Structures*, (2003), 241–254
19. Yevtushenko A.S. : System of data analysis "Concept Explorer", *Proceedings of the 7th national conference on Artificial Intelligence KII-2000*, (2000), 127–134

Consecutive-ones: handling lattice planarity efficiently[3]

Elaine M. Eschen[1], Nicolas Pinet[2], and Alain Sigayret[2]

[1] Lane Dept. of CSEE, West Virginia University (USA).
elaine.eschen@mail.wvu.edu
[2] LIMOS CNRS UMR 6158 - 63177 Aubière (France).
sigayret@isima.fr

Abstract. A concept lattice may have a size exponential in the number of objects it models. Polynomial-size lattices and/or compact representations are thus desirable. This is the case for planar concept lattices, which has both polynomial size and representation without edge crossing, but a generic process for drawing them efficiently is yet to be found. Recently, it has been shown that when the relation has the consecutive-ones property (i.e, the matrix of the relation can be rapidly reorderd so that the 1s are consecutive in every row), the number of concepts is polynomial and these can be efficiently generated. In this paper we show that a consecutive-ones relation $|\mathcal{R}|$ has a planar lattice which can be drawn in $O(|\mathcal{R}|)$ time. We also give a hierarchical classification of polynomial-size lattices based on structural properties of the relation $\mathcal{R}$, its associated graphs G_{bip} and $G_{\mathcal{R}}$, and its concept lattice $\mathcal{L}(\mathcal{R})$.[3]

Keywords: *consecutive-ones matrix, consecutive-ones relation, planar lattice, polynomial lattice, chordal-bipartite graph, Ferrers dimension.*

1 Introduction

There is a strong relationship between concept lattices and graphs, which enables to use the rich mine of graph results. For example, [4] presented a very efficient algorithm to generate the concepts when the relation has the consecutive-ones property. They used the natural association between a finite context $(\mathcal{O}, \mathcal{P}, \mathcal{R})$ and a bipartite graph $G_{bip} = (\mathcal{O} + \mathcal{P}, E)$, where $xy \in E$ iff $(x, y) \in \mathcal{R}$. Another interesting feature is that these consecutive-ones relation have a few $O(|\mathcal{R}|)$ concepts, and can be recognized and reorganized in very efficient $O(|\mathcal{R}|)$ time ([8]). This relationship was also illustrated by [5] who defined an encoding graph G_R which is the complement of G_{bip} (i.e., $G_{\mathcal{R}} = (\mathcal{O} + \mathcal{P}, F)$, $\mathcal{O}$ and $\mathcal{P}$ are cliques and $\forall x \in \mathcal{O}$, $\forall y \in \mathcal{P}$, $xy \in F$ iff $(x, y) \notin \mathcal{R}$). $G_{\mathcal{R}}$ was then used by [3] for generating all the concepts in the general case.

In [6], the problem of restricting a relation to a relation which has a polynomial number of concepts was addressed by suggesting to embed the relation

[3] This research has been developped on june 2007, while E. Eschen was invited at Clermont-Ferrand by University Blaise Pascal. Corresponding author A. Sigayret.

into a relation whose graph $G_{\mathcal{R}}$ is weakly chordal. In this case, the lattices $\mathcal{L}(\mathcal{R})$ have $O(|\mathcal{R}|)$ concepts.

To avoid handling exponential lattices, a $O(|\mathcal{O}+\mathcal{P}|)$ size substructure called Galois sub-hierarchy has been defined. There are several algorithms that compute the Galois sub-hierarchy in at least $O(|\mathcal{O}+\mathcal{P}|.|\mathcal{P}|^2)$ (see [1]).

Recently, [20] discussed about a characterization of the planar lattices as having a Ferrers dimension at most to two (see Theorem 1 below). Also concerned whith the number of concepts, he pointed out that these lattices have at most $O(|\mathcal{P}^3|)$ concepts. However, actually obtaining a planar drawing remains difficult.

Graph results enables us to remark that relations with Ferrers dimension at most two are a supercase of consecutice-ones relations, and a subcase of relations for which graphs G_{bip} are weakly chordal. This gives the idea that there is an interesting hierarchy of classes of concept lattices which need to be researched.

In this paper, we address two closely related problems:
• We discuss how to obtain an efficient planar drawing of the lattice for the sub-class of planar lattices whose relations have the consecutive-ones property.
• We study a hierarchical classification of polyomial size concept lattices, and give precise bibliographical references. This hierarchy combines results is based on Graph Theory, Order and Lattice Theory, and matrix patterning that should be helpful for further work on lattice representations.

2 Background

An undirected graph $G=(V,E)$ is said to be *chordal* (or *triangulated*) if it has no chordless cycle of length greater then 3. A graph G is said to be *weakly chordal* if it and its complement, $\overline{G}$, has no chordless cycle of length greater then 4. A *bipartite graph* is a graph $G=(V_1+V_2,E)$, where V_1 and V_2 are independent sets (i.e., each induces an edgeless subgraph). A *chordal-bipartite* graph is a graph that is bipartite and weakly chordal. The *neighborhood* of a vertex v in a graph $G=(V,E)$ is denoted and defined as $N(v)=\{x \in V | vx \in E\}$. A bipartite graph is a *chain graph* if for each V_i $i \in \{1,2\}$, the neighborhoods of vertices of V_i can be totally ordered by set containment (i.e., for any pair of vertices $u,v \in V_i$, either $N(u) \subseteq N(v)$ or $N(v) \subseteq N(u)$; equivalently, the graph has no induced $2K_2$).[4]

A relation $\mathcal{R} \subseteq (\mathcal{O} \times \mathcal{P})$ is said to be a *Ferrers relation* if $\forall x_1, x_2 \in \mathcal{O}$, $\forall y_1, y_2 \in \mathcal{P}$, $(x_1,y_1) \in \mathcal{R}$ and $(x_2,y_2) \in \mathcal{R}$ implies $(x_1,y_2) \in \mathcal{R}$ or $(x_2,y_1) \in \mathcal{R}$. The *Ferrers dimension* of a relation $\mathcal{R}$ is the minimum number of Ferrers relations the intersection of which is $\mathcal{R}$. The *chain dimension* of a bipartite graph is the minimum number of chain graphs the intersection of which yields the graph. Therefore, the Ferrers dimension of a relation $\mathcal{R}$ is equal to the chain dimension of G_{bip}. A *chain graph* is a graph with chain dimension 1; thus $\mathcal{R}$ is a Ferrers relation iff G_{bip} is a chain graph.

[4] A $2K_2$ of an undirected graph G is a quadruple of vertices x,y,z,t such that xy and zt are the only edges of G whose enpoints both are in {x,y,z,t}.

Planar lattices ar characterized by their Ferrers dimension:

Theorem 1. *([2],[7],[11]) The following are equivalent:*
– The Ferrers dimension of $\mathcal{R}$ is at most 2;
– $\mathcal{L}(\mathcal{R})$ has a planar representation;
– The order dimension of $\mathcal{L}(\mathcal{R})$ is at most 2;
– $\mathcal{L}(\mathcal{R})$ has a conjugate order.

When $G_{\mathcal{R}}$ is chordal ([5]), the corresponding matrix of a $\mathcal{R}$ presents a very specific pattern, called a *staircase*, as the rows are totally ordered by inclusion (see an example in Figure 1). We then have:

Theorem 2. *The following are equivalent:*
– $\mathcal{R}$ is a Ferrer relation;
– $\mathcal{R}$ has a staircase matrix;
– G_{bip} is a chain graph;
– $G_{\mathcal{R}}$ is chordal;
– $L(\mathcal{R})$ is a chain.

	a	b	c	d	e	f
1	×	×	×	×	×	
2	×	×	×			
3	×	×				
4	×	×				
5	×					

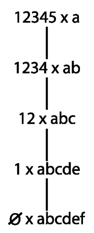


Fig. 1. A staircase matrix of a relation, and its concept lattice which is a chain.

3 Consecutive-ones lattices

Consecutive-ones lattices are planar. A relation is said to have the *consecutive-ones property (for rows)* if the columns of its binary matrix can be ordered such that in each row all the ones are consecutive. Figure 2 shows such a relation and the corresponding concept lattice. Planarity of consecutive-ones lattices is a direct consequence of Theorem 1 applied to the following observation.

Observation 3 *If a relation has the consecutive-ones property, then its Ferrers dimension is at most 2.*

Proof: Let M be a consecutive-ones matrix of a relation $\mathcal{R}$. Let $\mathcal{F}_1$ be the relation obtained from M by: for each row, changing to one each zero that occurs before the sequence of ones in the row. Let $\mathcal{F}_2$ be the relation obtained from M by: for each row, changing to one each zero that occurs after the sequence of ones in the row. By Theorem 2, $\mathcal{F}_1$ and $\mathcal{F}_2$ are Ferrers relations, and clearly $\mathcal{R} = \mathcal{F}_1 \cap \mathcal{F}_2$. Thus, the Ferrers dimension of $\mathcal{R}$ is at most 2. $\diamondsuit$

Note that Observation 3 is also implied by a characterization of a larger class of relations (see [15]), those of Ferrers dimension 2 for which $\mathcal{F}_1 \cup \mathcal{F}_2$ is complete (which corresponds to the interval digraphs).

	a	b	c	d	e	f
1	×	×	×	×	×	
2	×	×	×			
3	×	×				
4	×					
5		×	×	×		
6		×				
7				×	×	×
8				×	×	
9				×		

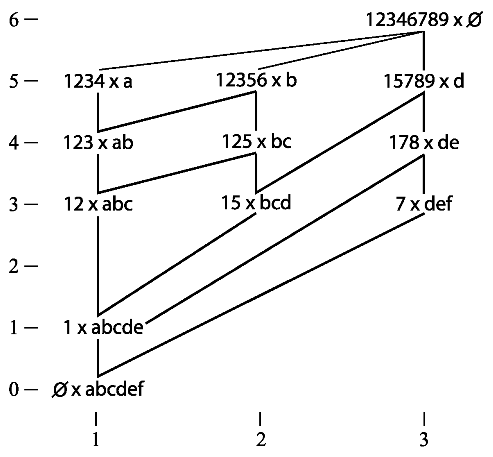


Fig. 2. A consecutive-ones relation and the associated concept lattice. In the matrix, objects and properties are ordered such that in each row all the 1s are consecutive. The margin of the Lattice drawing gives the coordinates of the concepts, as computed by a execution of algorithm PlaCoL (see below).

All the planar lattices do not have the consecutive-ones property, as illustrated in Figure 3. In the matrix of this example the columns can not be reordered to obtain a consecutive-ones matrix: columns *a* and *b* must be consecutive (because of object 1) and so do columns *b*, *c*, and *d* (because of object 3); thus no permutation of the columns can erase the hole on row 5.

	a	b	c	d	e
1	×	×			
2	×				
3		×	×	×	
4			×		
5		×			×

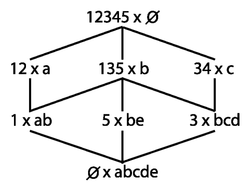


Fig. 3. A relation which has not the consecutive-ones property, and the corresponding planar lattice.

Unfortunately, the Ferrers dimension does not give much insight towards an efficient generic process for generating a planar drawing. We now present an algorithm, which we call PlaCoL (for *PLAnar drawing of Consecutive-Ones Lattices*), that use the specificity of a consecutive-ones matrix to efficiently build a planar representation of the lattice. With a slight modification, this algorithm computes the Galois sub-hierarchy.

The algorithm. We use as input a consecutive-ones matrix in which the rows are ordered by increasing value of starting column SC. The rows sharing the same SC are ordered by decreasing value of ending column EC; thus, defining one of the staircases s of the matrix. Note that several rows may have the same EC value and the same SC value. The algorithm contructs for each staircase a chain of the lattice each concept of which is associated with a row of the matrix (see [4] for more explanation on this process). Tables HEAD and TOP memorize the concepts from which an edge has to be drawn respectively to a concept of a later chain (lateral edge) or to the top. The edges from the bottom to the atoms are drawn online. For example, on the matrix M of Figure 2, PlaCoL first generates the bottom $\emptyset \times \mathcal{P}$ as $M[1, f] = 0$. Staircase 1 (rows 1-4) successively generates the elements of the first chain $1 \times abcde \sim 1234 \times a$. Staircase 2 (rows 5-6) generates chain $15 \times bcd \sim 12356 \times b$ and the lateral edges outgoing the first chain. Staircase 3 (rows 7-9) generates chain $7 \times def$ ~top and the lateral edges outgoing staircase 2. PlaCOL ends by drawing the incoming edges of the top. The position of each concept is determined by its chain number (abscissa) and its intent size (ordinate) as described below.

```
Algorithm PlaCoL
Input: A consecutive-ones matrix in the form described above.
Output: A planar drawing of the corresponding concept lattice.
Process:
If the first row of the matrix ended with a zero then
    Generate bottom = $\emptyset \times \mathcal{P}$;
    ABSCISSA(bottom)← 1; ORDINATE(bottom)← 0;
    Create an egde from bottom to the first concept generated next;
// else the first concept generated next will be the bottom.
For each staircase $s$ do:
    Let $t$ be the first row of $s$;
    For each list $R$ of rows of $s$ sharing the same EC do:
        Let $r$ be the last row of $R$;
        If $\exists i \in [t \,..\, r] \mid$ SC($i$)=SC($s$) then
            Let $u$ be the first such row $i$;
            If SC($r$)<SC($s$) then $A \leftarrow \emptyset$ else $A \leftarrow [u \,..\, r]$;
            $B \leftarrow$[SC($s$)..EC($r$)];
            Create concept $A \times B$;
            TOP($s$)← $A \times B$;
            ASBCISSA($A \times B$)←$s$;
            ORDINATE($A \times B$)←$|\mathcal{P}| - |A|$; // i.e., 0 or $(r - u)$
            If $A \times B$ is not the first concept generated for staircase $s$ then
                // Edge between consecutive concepts of same chain.
                Create an edge from the previously generated concept to $A \times B$;
            If $\exists i \in R \mid$ SC($i$)<SC($r$) then // Create a lateral edge.
                Let $q$ be the last such row in $R$, $h$ the corresponding staircase;
                Create edge from HEAD($q$) to $A \times B$;
                TOP($h$)←$A \times B$;
            else if $A \times B$ first concept of $s$ and extent(bottom) empty then
                Create an edge from bottom to atom $A \times B$;
        If $s$ is not the last staircase and EC($r$)≥SC($s+1$) then
            Insert rows $R$ in staircase $s+1$;
            // preserving increasing order on EC, and on SC.
            HEAD($r$)←$A \times B$;
        If $r$ is the last row of the last staircase then
            If the intent of the last generated concept is $\mathcal{P}$
                then this concept is the top element;
                else generate the top concept $\mathcal{O} \times \emptyset$;
For each staircase $s$ do:
    Create an edge from TOP($s$) to the top element.
```

We first prove that Algorithm PlaCoL actually generates the concepts in an order compatible with a planar drawing.

Lemma 4 *Algorithm* PlaCoL *constructs successive chains (first one with bottom, last one with top, each chain labeled by its generating staircase), which yields a partition of the lattice in which a concept is generated only after all its ancestors have been generated.*

Proof: The first staircase, $s = 1$, generates all the concepts that contain the first property SC(1) (corresponding to the first column). By Theorem 2, these concepts form a chain of concepts ordered by increasing size of extent $[t \mathinner{..} r]$. By construction of the matrix, all the concepts generated afterwards will not have SC(1) in their intents, and thus, will not be ancestors of any concept of the first chain; thus the first generated chain is actually a chain of the lattice containing exactly those concepts with SC(1) in their intent. Recursively, each staircase s generates the concepts the intents of which contain SC(s) and not SC(1) through SC($s-1$); this corresponds to a chain in which each concept will not be an ancestor of a previously generated concept. As each concept is generated once (see [4]), this yields a partition of the lattice into chains; i.e., a linear extension which respects the claimed property on concepts. $\Diamond$

We now know that the only edges that can be created in Algorithm PlaCoL will be edges going up from a given concept to another concept higher either in the same chain or laterally to a concept generated by a later staircase. There remains to prove that these lateral edges are non-crossing edges.

Theorem 5. *Assume an execution of Algorithm* PlaCoL *on a consecutive-ones matrix, resulting in a corresponding concept lattice diagram. If the diagram has an edge from a concept C_s of generated chain s to a concept C_u of generated chain $u \neq s$, then $u > s$ and this induces no crossing edge that is impossible to avoid.*

Proof: Let C_s and C_u be concepts respectively generated by two different staircases s and u. Suppose the diagram contains an edge e from C_s to C_u.
– By Lemma 4, $s > u$ is impossible.
– If $u = s + 1$, there is no obstacle.

– If $u > s + 1$ there will be at least one intermediate chain t. Let S be the first concept generated by chain s and S' its last concept. We similarly define T, T', U, and U', for chains t and u. Four exclusive situations could occur, which are illustrated by Figure 4:
Case 1. e is a transitivity edge and therefore is not in the diagram of $\mathcal{L}(\mathcal{R})$. Otherwise, e is not a transitivity edge: next cases.
Case 2. T is a descendent of C_s – i.e., there is an edge S''–T with S'' descendent of C_s in chain s. Thus edge e can be drawn below edge S''-T and there is no crossing. Otherwise, e is forced to cross some non-transitivity edge of an intermediate chain t: next cases.
Case 3. C_u is not a descendent of any concept of t. Note there may be some edge from a concept T'' of t to a concept U'' of u which is a descendent of C_u, but it does not matter. Since C_u is not a descendent of T, the rows of T (i.e., the last objects in its extent) have not been used to construct C_u, i.e., EC(T)<EC(C_u). As we are not in case 2, T is not a descendent of C_s (nor an ancestor) and then EC(T)>EC(C_s). As a consequence EC(C_s)<EC(C_u) which contradicts the existence of the edge e.
Case 4. Otherwise, C_u is a descendent of some concept C_t of t. Let I = intent(C_s) $\cap$ intent(C_t). Since C_u is a descendent of both C_s and C_t, we have intent(C_u) $\subset I \neq \emptyset$. If I=intent(C_t), then intent(C_t)$\subset$intent(C_s) and C_t is a descendent of C_s. Then e is a transitivity edge. If $I \subset$ intent(C_t), there must exist a concept X such that intent(X) $= I$, X is a descendent of C_s and C_t, and X is on chain t. Since, intent(C_u) $\subset I =$ intent(X), C_u is a descendent of X. Then e is a transitivity edge. This last case is thus impossible. $\diamondsuit$

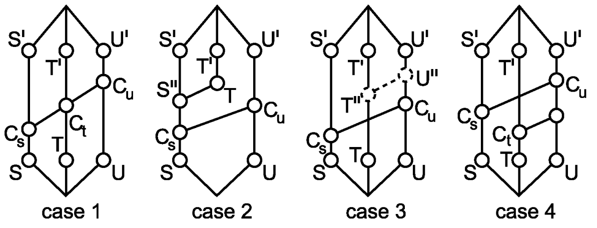


Fig. 4. Different situations for the proof of Theorem 5.

We now prove that all the edges of the diagram are actually provided by Algorithm PlaCoL. For this we need the following lemma:

Lemma 6 *In the diagram of a consecutive-ones lattice, each concept different from top has at most one incoming edge from concepts of a previous chain generated by Algorithm* PlaCoL.

Proof: Suppose the diagram has both edge C_s–C_u and edge C_t–C_u. If C_s, C_t, and C_u be concepts generated by three different staircases $s \leq t \leq u$. This corresponds exactly to the case number 4 of Theorem 5, and thus, is impossible.

If C_s and C_t are generated by the same staircase, and thus, are in the same chain. W.l.o.g. C_s is an ancestor of C_t. As a consequence, edge C_s–C_u is a transitivity edge. $\diamondsuit$

Theorem 7. *Algorithm* PLACOL *provides all the edges of the diagram.*

Proof: Recall, by Lemma 4, PLACOL partitions the concepts of the lattice in to a collection of chains. Suppose the diagram contains an edge $e{=}C_1$–C_2. Either C_1 and C_2 are in the same chain of this partition or not. Concepts C_1 and C_2 are generated by the same staircase iff they are in a same chain; then e is created by the algorithm iff C_1 and C_2 are consecutive. If C_1 and C_2 have been generated by two different staircases i and j respectively, then $i < j$, and this means that rows of staircase i have been inserted in the following staircases until (at least) staircase j. Algorithm PLACOL detects whether there exists such an insertion and, if yes, selects the most recent such staircase q (i.e., of highest index) and the corresponding concept, which is given by HEAD. By Lemma 6, such a situation occurs at most once for each concept and the algorithm creates a lateral edge. Finally, each highest concept of a chain will be linked to the top, unless this highest concept is an ancestor of a concept of another chain. TOP ensures there will be no created edge between top and a concept that has been proven to be a ancestor of some concept of a later chain. Thus, each edge of the diagram is generated exactly once. $\diamondsuit$

Complexity analysis. As a consecutive-ones lattice has at most $O(|\mathcal{R}|)$ concepts ([4]) and, by Lemma 6, the diagram of a consecutive-ones concept lattice has at most $O(|\mathcal{R}|)$ edges, each of them is generated without extra cost. Thus, Algorithm PLACOL has the same $O(|\mathcal{R}|)$ complexity as Algorithm CONS-1.

Drawing the diagram. The above considerations do not ensure a correct planar drawing of the lattice, as some pair of edges that could be drawn without crossing may cross in an incautious drawing. Fortunately, this problem can be solved by choosing for each concept C an ordinate value $y(C)$ that is a function of the size of its intent: $y(C) = |\mathcal{P}| - |\text{intent}(C)|$. As intent$(C)$ is an interval, its size is computed in constant time. Note that the ordinates can be computed using the extent instead of the intent, especially when Algorithm PLACOL is modified to give the whole extent label (see below).

Furthermore, each edge is drawn when its ending point is reached and, at this time, its starting point is memorized in HEAD; there is thus no need to memorize all the previously generated concepts – an expensive constraint for many lattice generating algorithms.

The abscissa of a concept is given by the number of the chain to which this concept belongs. With this, the bottom's abscissa is 1 and the top's abscissa is the number of staircases in the input matrix. We may chose several values both for the initial abscissa and for the increment.

If we want to set bottom and top on the same vertical line, all the coordinates can be rotated accordingly, using a simple mathematical formula: if (x_0, y_0) are the coordinates of bottom and (x_1, y_1) the coordinates of top, we will rotate all the coordinates by angle $\theta = ArcSin((x_1 - x_0)/\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2})$. Consequently, the previously verticality will have a new $\frac{\pi}{2} + \theta$ direction; the other edges, which had a direction in interval $]0, +\frac{\pi}{2}[$, will have a direction in $]\theta, \theta + \frac{\pi}{2}[$, which remains correct because $0 < \theta \leq +\frac{\pi}{2}$. Other changes to coordinates may be chosen, as desirable for different purposes; in particular, for the respect of attribute-additivity (see [14]).

Concept labeling. In order to obtain a time complexity of $O(|\mathcal{R}|)$, Algorithm CONS-1 only labels the intent of each concept. A complete labeling of the extent would have an elementary cost corresponding to the size of each generated extent. This is due to the fact that a consecutive-ones matrix ensures each *intent* is an interval, but this is not true for the *extents*, as a concept created by a set R of rows of a staircase u can use rows of a previous staircase s without using any row of some intermediate staircase t (in other words, some row of s has an EC greater or equal to the EC of the rows in R, but no row of t does).

It is possible to memorize and update a list of the rows whose ECs are not outdated, and this can be done in $O(|\mathcal{O}|)$ per concept. Consequently, the total complexity of Algorithm CONS-1, as well of Algorithm PLACOL, would be $O(|\mathcal{O}|.|\mathcal{R}|)$, if we want a complete labeling of the extents.

On the other hand, for lattice drawing, the concepts need to be labeled only by the introduced objects and/or properties. Then, in Algorithm PLACOL we chose to label the intent with the introduced objects. Computation of a concept C is determined by the set R of rows defined within a given augmented staircase s by a common value of EC. This will determine the introduced objects of intent(C). Intent(C) is partitioned into three sets. The first set corresponds to the rows that have been inserted in s and inherited from a previous staircase; they are characterized by a SC value different than SC(s). The second set corresponds to the rows that have been used to create a previous concept of the same chain; they are characterized by a EC bigger then the smallest EC of R. The last set corresponds to the rows that have not been previously used. The objects introduced by C are exactly the ones of the third set, which are computed with no extra-cost by Algorithm PLACOL.

We can provide the corresponding introducer-labeling for the properties, making a few slight changes in Algorithm PLACOL; this will have no impact on the complexity. As all the rows of the same staircase share the same SC and are ordered by decreasing value of EC, only the last concept of a chain may introduce a property. As the concepts of a chain have no descendent in a previously generated chain, a concept cannot introduce a property that is in the intent of some concept of a chain that is generated later. Thus, the last concept of each chain s is the introducer of properties in interval $[\text{SC}(s) \mathinner{..} \text{SC}(s+1)[$, except for the last chain l (in which the last concept introduces $[\text{SC}(l) \mathinner{..} \text{EC}(l)]$). If z is the first column of the matrix with no 1, the properties of $[z \mathinner{..} |\mathcal{P}|]$ are introduced by bottom. Each property p of the remaining interval $]\text{EC}(l) \mathinner{..} z[$ is introduced by the last concept of the last chain whose row's EC is bigger than p; this can be determined in a lazy way by preprocessing the column that labels each staircase with the appropriate list of properties from $]\mathrm{EC}(l) \mathinner{..} z[$.

Galois sub-hierarchy. As a consequence, Algorithm PLACOL can also be modified to compute the Galois sub-hierarchy: using the introducer-labeling, all the concepts whose intent and extent are both empty will be dismissed. The remaining concepts are the elements of the sub-hierarchy. When a concept is dismissed, it is replaced by its highest maintained ancestor in the same chain for the outgoing edges and by its lowest maintained descendent in the same chain for the incoming edges; bottom and top are managed accordingly. Thus, using elementary techniques, the total complexity remains the same: the Galois sub-hierarchy is computed in $O(|\mathcal{R}|)$ time, which is better than the generic algorithms ([1]), and should be refinable to $O(|\mathcal{O} + \mathcal{P}|)$.

Figure 5 give a planar drawing of the lattice of Figure 2 with the standard labeling of the concepts.

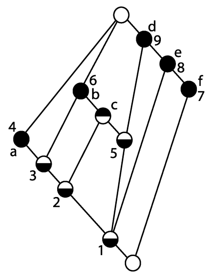


Fig. 5. A standard planar drawing of the lattice of Figure 2.

4 A hierarchy of polynomial concept lattices

In this section, we present a hierarchical classification of polynomial-size concept lattices using a known hierarchy of bipartite graph classes: {chain} ⊂ {biconvex} ⊂ {convex} ⊂ {ATE-free} ⊂ {chordal-bipartite} ⊂ {bipartite} (for more information on these graph classes, see [19], [9], and [10][5]). This classification shows that studying the length of cycles in graphs ([12]) is important for concept lattices.

1. The smallest class corresponds to the lattices that are simply a chain, as in Figure 1. This corresponds to the situation of Theorem 1.

2. The second class corresponds to matrices that have the consecutive-ones property for the objects as well as for the properties.

[5] Note the use of 'bichordal' for chordal-bipartite, 'bigraph' for bipartite graph, 'digraph' for directed graph.

	Matrix M	Relation $\mathcal{R}$	Bipartite graph G_{bip}	Co-bip. graph $G_{\mathcal{R}}$	Lattice $\mathcal{L}(\mathcal{R})$
1	staircase	Ferrers dim. = 1	chain graph	chordal	chain, size $O(\lvert\mathcal{O}\rvert)$
2	doubly consecutive-ones		biconvex		
3	consecutive-ones		convex		size $O(\lvert\mathcal{R}\rvert)$
4	see [10]	Ferrers dim. ≤ 2	chain dim. ≤ 2	circular-arc	planar
5			ATE-free		
6	Γ-free		chordal-bipartite	weakly chordal	
7					polynomial size

Fig. 6. The hierarchy of polynomial-size concept lattices. The smaller class is above.

3. A wider class corresponds to the consecutive-ones property; as shown above, the corresponding lattices are planar. G_{bip} is convex.
4. The next class is the class of planar lattices. They are characterized by Theorem 1, which is difficult to handle (see e.g., [20]). [10, 17, 18] indirectly give another characterization of planar lattices by setting an equivalence between a Ferrers dimension ≤ 2 relation and a pattern in the matrix. They also prove that the graphs G_{bip} in this case are exactly the interval containment bipartite graphs. In [17, 16], the graphs $G_{\mathcal{R}}$ are characterized as circular-arc co-bipartite graphs.
5. The previous class of bipartite graphs is properly contained in a subclass of chordal-bipartite graphs, the class of ATE-free chordal-bipartite graphs (see [10]).
6. In [6], it is proved that a concept lattice for which G_{bip} is chordal-bipartite (equivalently, weakly chordal and bipartite) has $O(|\mathcal{R}|)$ concepts. All the previous classes we present are included in this one, and thus, have polynomial-size concept lattices. Testing whether a graph is chordal-bipartite can be done in $min\{O(|\mathcal{O}+\mathcal{P}|^2), O(|\mathcal{R}|.log(|\mathcal{O}+\mathcal{P}|))\}$ time (see e.g., [19] for a discussion), and this can be performed directly on the matrix M. These graph algorithms compute a doubly lexical ordering of the matrix and search for a specific forbidden pattern called a Γ[6]. Every doubly lexical ordering has no Γ iff the corresponding graph is chordal-bipartite. We know how important is the use of total orders on objects or properties for concept generation: arbitrary order such as lectic (i.e. lexical) order of [13] or structural order as domination of [3].
7. Our largest class is the one of polynomial size concept lattices.

5 Conclusion

For a relation which has a planar representation, we have provided new insight which can help insert it into a hierarchy of relation classes; in particular, we have shown that the subclass of consecutive-ones lattices are endowed with easy drawing algorithms.

Moreover, any given binary relation can easily be embedded into a consecutive-ones relation by adding or removing 1s, thus restricting the number of concepts

[6] For h<i, j<k, M[h,j]=M[h,k]=M[i,j]=1 and M[i,k]=0 forms a Γ of matrix M.

to a tractable (polynomial) number, as well as yielding a fast algorithm for drawing a planar representation. However, we do not know how to do this efficiently while adding or removing a minimum or even minimal set of 1s, a question we leave open.

Another question would be to relax the planarity condition to allow an 'almost planar representation'. As discussed in this paper, it turns out that planar lattices correspond to bipartite graphs with no cycles of length strictly more than four. It would be interesting to relax this condition to allow cycles of length six but not more.

References

1. Arévalo G., Berry A., Perrot G., Sigayret A.: *Performances of Galois sub-hierarchy-building algorithms.* ICFCA'07, LNCS/LNAI 4390, 2007, pp.166-180.
2. Baker K.A., Fishburn P. Roberts F.S.: *Partial orders of dimension 2.* Networks 2, 1971, pp.11-28.
3. Berry A., Bordat J-P., Sigayret A.: *Concepts can't afford to stammer.* INRIA Proc. JIM'03 (Metz, Fr), 2003. To appear in AMAI as *A local approach to concept generation.*
4. Berry A., McConnell R.M., Sigayret A., Spinrad J.P.: *Very fast instances for concept generation.* ICFCA'06, LNAI 3874, 2006, pp.119-129.
5. Berry A., Sigayret A.: *Representing a concept lattice by a graph.* Discrete Applied Mathematics, 144(1-2), 2004, pp.27-42.
6. Berry A., Sigayret A.: *Obtaining and maintaining polynomial-sized concept lattices.* FCAKDD, Proc. ECAI'02 (15th Eur. Conf. on Artificial Intelligence), 2002, pp.3-6.
7. Birkhoff G.: *Lattice Theory.* AMS, 3rd edition, 1967.
8. Booth S., Lueker S.: *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms.* J.compute.Syst.Sci. 13, 1976, pp.335-379.
9. Brandstädt A., Le V.B., Spinrad J.P.: Graph Classes: a Survey. SIAM Monographs on Discrete Mathematics and Aplications, 1999.
10. Das A., Sen M.: *Bigraphs/digraphs of Ferrers dimension 2 and asteroidal triple of edges.* Discrete Mathematics, 295:1-3, 28 may 2005, pp.191-195.
11. Dushnik B., Miller E.W.: *Partially ordered sets.* Am.J.Math. 63, 1941, pp.600-610.
12. Eschen E.M., Sritharan R.: *A Characterization of Some Graph Classes with No Long Holes.* J. Comb. Theory, Series B 65, 1995, pp.156-162.
13. Ganter B.: *Two basic algorithms in concept analysis.* Preprint 831, Technische Hochschule Darmstadt, 1984.
14. Ganter B., Wille R.: *Formal Concept Analysis.* Springer, 1999.
15. Lin J.I., Sen K.M., West B.D.: *Classes of interval digraphs and 0,1-matrices.* 28th S.E. Conf.Comb.Graph.Th. and Congr.Numer., 1997, pp.201-209.
16. Sen M., Basu A., Das S.: *Bigraphs of Ferrers dimension two and circular arc graphs.* Manuscript 2003.
17. Sen M., Das S., Roy A.B., West D.B.: *Interval digraphs: an analogue of interval graphs.* J. Graph Theory 13, 1989, pp.189-203.
18. Sen M., Sanyal B.K., West D.B.: *Representing digraphs using intervals and circular arcs.* Discrete math. 147, 1995, pp.245-253.
19. Spinrad J.P.: Efficient Graph Representations. Fields Inst. Monographs, American Mathematical Society, Rhode Island, 2003.
20. Zschalig C.: *Bipartite Ferrers-graphs and planar concept lattices.* ICFCA'07, LNAI 4390, 2007, pp.313-328.

CLANN: Concept Lattice-based Artificial Neural Network for supervised classification

Norbert Tsopzé[1,2], Engelbert Mephu Nguifo[2], and Gilbert Tindo[1]

[1] Université de Yaoundé I, Faculté des Sciences,
Département d'Informatique BP 812
Yaoundé - Cameroun
[2] CRIL - CNRS, IUT de Lens,
SP 16, Rue de l'Université, 62307 Lens Cedex, France

Abstract. Multi-layer neural networks have been successfully applied in a wide range of supervised and unsupervised learning applications. As they often produce incomprehensible models they are not widely used in data mining applications. To avoid such limitations, comprehensive models have been previously introduced making use of an apriori knowledge to build the network architecture. They permit to neural network methods to deserve a place in the tool boxes of data mining specialists. However, as the apriori knowledge is not always available for every new dataset, we hereby propose a novel approach that generates a concept semi-lattice from initial dataset, to directly build the neural network architecture. Carried out experiments showed the soundness and efficiency of our approach on various UCI.

1 Introduction

Neural network (also called connexionnist network) technique is one of the most used techniques in machine learning. Feed-forward neural networks have been applied to solve many problems: handwritten caracters recognition, molecular biology, etc. There exists a plethora of architectures and algorithms about neural networks. But it is very difficult to choose those which are the best for a given task [7]. Finding the architecture of the network to be used for solving a given problem is a very complicated task. In fact there is no existing exact method for defining the number of layers, the number of neurons in each layer and the connections between neurons [7]. Define the topology of neural network consists of answering certain questions: how many hidden layers? how many neurons per layer? which connection policy? how to define each unit threshold? Moreover, only the decision aspect of the neural model (which consists in using the model in the decision task) is used and no importance is given to its descriptive aspect (how decision is taken). Besides, the intelligibility of learned model is fundamental in datamining [3]; the absence of an explanation capability limits its use [1]. Black box neural network is usually not comprehensible. In [1] authors enhance the transparency of the network in the decompositional extraction rules process.

The topology becomes important for the intelligibility of the neural model and in the process of extracting rules from neural network model.

When tackling a supervised classification problem, the widely used approach (ad'hoc) to define a feed-forward network topology consists in using three layers: input, hidden, output. The number of neurons of the input layer is the number of variables or attributes. The number of neurons of the output layer is the number of classes. The number of neurons in the hidden layer is calculated as the mean of the number of input units and the output units. In the literature, different approaches have been reported to build the network topology. These research works could be divided into two groups:

1. Search an optimal network to minimize the number of units in the hidden layers [11]. This technique brings out a constructive solution to the problem without an apriori knowledge. Its main limitation is the intelligibility of the resulting network, the network is a black box i.e. no semantic is associated to each node.
2. Use a set of apriori knowledge on the problem domain and build neural network from this knowledge [13]. The main advantage here is that the result is a comprehensive network i.e. each node in the network represents one variable in the rules set and each connexion between two nodes represents one dependence between variables. But it is not possible to use it when the apriori knowledge is not available.

The aim of this paper is to introduce CLANN, an approach based on concept lattices which goes beyond the limitations of the existing approaches. Concept lattices is an ordered graph composed by formal concepts. Given an input matrix (also called formal context) specifying a set of objects and their corresponding properties, a formal concept is a pair containing both a subset of objects (X) and a subset of properties (Y), such that Y is the set of all properties shared by the objects of X, and X is the set of all objects that share the properties of Y. In fact concept lattices has been extremely used in supervised classification [10]. The use of the concept lattices [5] in classification proceeds in two phases: the first phase called the training phase consists in selecting interesting concepts among the set of concepts or in extracting rules. Selected concepts and/or rules are then used to take decision about new objects in the second phase [10].

CLANN uses directly the Hasse diagram of the built semi-lattice to define the architecture of the neural network. Within data mining domain, concepts and connexions defined in this diagram are generally used to extract rules into the formal context. Here concepts and connexions are used to define neurons and their connexions. We believe that this approach can be helpful when the apriori knowledge is not available.

Many constraints (or heuristics) presented in the literature can be used to prune the concept lattices in order to reduce the number of generated concepts. The CLANN approach has two main advantages: first it finds a comprehensive model when the knowledge is not available, secondly it is possible to justify the built neural network topology. Another advantage of this approach is the possibility to easily extract rules from the network. Carrying out experimentations of this method using benchmarks from UCI repository shows interesting results in terms of precision, and also that CLANN can be better than many comprehensive models such as $C4.5$ and $IB1$.

The rest of this paper is organized as follows: the next section presents some notions used in Formal Concept Analysis (FCA). The third section presents the CLANN approach. Experimentation and obtained results are described in section four. Related works about extracting rules in trained neural networks end this paper.

2 PRELIMINARIES

2.1 FCA basic notions

This section presents some important notions used in FCA. More notions about FCA could be found in [5].

A **formal context** is a triplet $C = (O, A, I)$ where O is a non empty finite set of objects, A a non empty finite set of attributes (or items) and I is a binary relation between elements of O and elements of A (formally $I \subseteq \{(o, a) \,/ o \in O \text{ and } a \in A\}$). A context C can be represented as binary matrix M (where $M_{i,j} = 1 \iff (o, a) \in I, 0$ otherwise) or transactions database (a row o is the collection of attributes which are verified by the object o). A(resp. O) is also called set of items (resp. transactions). Table 1 is an example of context. We denote in the next sections the set $\{a, b, c, d\}$ $(resp. \{1, 2, 3, 4\})$ as abcd (resp.1234). The objects in table 1 could be divided into two parts: a set of examples O^+ (for instance $O^+ = \{1, 2, 3, 4, 5, 6\}$ for the context table 1) and the set of counterexamples O^- (for instance $O^- = \{7, 8, 9, 10\}$ for the context table 1).

Table 1. Example of context presented as boolean matrix; two-class (+ and -) data indicated by class column.

Objects	a	b	c	d	e	f	class
1	1	1		1	1	1	+
2	1		1		1	1	+
3		1	1	1	1		+
4		1	1	1			+
5	1	1		1	1		+
6	1		1		1		+
7		1		1		1	-
8		1			1	1	-
9			1		1	1	-
10	1			1	1		-

Let f *and* g be two applications defined as follows: $f : 2^O \longrightarrow 2^A$, such that $f(X) = X' = \{a \in A \ / \ \forall o \in X \ , \ (o, a) \in A\}$, $X \subseteq G$ and $g : 2^A \longrightarrow 2^O$, such that $g(Y) = Y' = \{o \in O \ / \ \forall a \in Y \ , \ (o, a) \in I\}$, $Y \subseteq A$; a pair (X, Y) is called **concept** iff $X = Y'$ *and* $Y = X'$. X (resp. Y) is called **extension** (resp. **intention**) of the concept, an intention of concept is also called description.

Let (X_1, Y_1) and (X_2, Y_2) be two concepts, and $\leq$ a relation defined on a entire set of concepts extracted from the context. $(X_1, Y_1) \leq (X_2, Y_2) \ if \ X_1 \subseteq X_2 \, (Y_1 \supseteq Y_2)$. (X_1, Y_1) is called **successor** of (X_2, Y_2) and (X_2, Y_2) **predecessor** of (X_1, Y_1) if there is no concepts between them. The relation $\leq$ defines an order relation on the entire set L of concepts [5]; the set of all concepts L with the order relation $\leq$ define the concept lattices.

A lattice is a partially ordered set (or poset) in which every pair of elements has a unique supremum (the elements' least upper bound; called their join) and an infimum (greatest lower bound; called their meet). A semilattice is a partially ordered set (poset) closed under one of two binary operations, either supremum (join) or infimum (meet). Hence we speak of either a join-semilattice or a meet-semilattice. If an ordered set is both a meet- and join-semilattice, it is also a lattice.

2.2 Supervised classification

In a supervised classification process, the system works into two phases [10]: training (learning) phase and testing (evaluation) phase. The dataset is separated into two subsets, the first subset is used to build and train the model during the training phase while the second is used to evaluate the model during the testing phase. For instance, the dataset of table 1 could be divided into two subsets $\{1, 2, 5, 6, 9, 10\}$ as training set and $\{3, 4, 7, 8\}$ as test set. In the literature, different techniques were reported for supervised classification problem [7]: instance-based learning, decision trees, artificial neural network, support vector machine, bayesian network, ... We are here concerned with artificial neural network.

2.3 Feed-forward neural networks

Neural networks is a set of interconnected neurons (also called units), which exchange information with one to another and communicate with the external environment. Different type of neural networks were reported in the literature [2]: the feed forward neural networks (perceptron, multilayer perceptron, ...) and the reverse feed forward neural networks (for instance the "Adaptive resonance theory" networks). More notions about artificial neural networks can be found in [2]. A feed forward neural network is composed of different layers:

1. The input layer is composed of neurons which receive information from outside.
2. The internal layer is composed of units which make intermediate treatment; it could be composed of many layers.

3. The output layer is formed by units which make decision.

A neuron is a process unit which has an internal memory, communicates with the external environment. Its state is defined as a function of its inputs; this function is called activation function [2]. The connexions between neurons define the architecture of the network. The neuron can also be called unit, it is active if its internal state value is 1 and inactive if this value is 0. After defining the network topology, the connexion weights between units are trained by backpropagation [12]. The connexion weights between two units a and b define the effect of the neuron a on the neuron b.

3 CLANN APPROACH

We describe in this section the different steps of our new approach as shown by figure 1. The process of finding the architecture of neural networks are three-folds: (1) build a join semi-lattice of formal concepts by applying constraints to select relevant concepts; (2) translate the join semi-lattice into a topology of the neural network, and set the initial connections weights; (3) train the neural network.

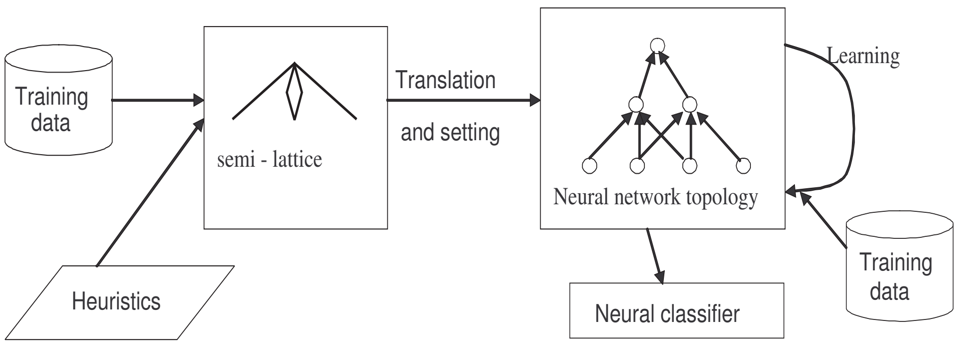


Fig. 1. Neural network topology definition.

3.1 Semi-lattice construction

There are many algorithms [8] which can be used to construct concept lattices; few of them build the Hasse diagram. Lattice could be processed using top-down or bottom-up techniques. In our case, a levelwise approach presents advantage to successively generate concepts of the join semi-lattice and the Hasse diagram. For this reason, we choose to implement the Bordat algorithm [8] which is suitable here. Concepts included in the lattice are only those which satisfy the defined constraints.

Algorithm 1 Modified Bordat algorithm

```
Require: Binary context $K$
Ensure: concept lattices (concepts extracted from $K$) and the Hasse diagram of the
    order relation between concepts.
 1: Init the list $L$ of the concepts $(G, \{\})$ $(L \leftarrow (G, \{\}))$
 2: repeat
 3:   for concept $c \in L$ such that his successors are not yet been calculated do
 4:     Calculate the successors $c'$ of $c$.
 5:     if the specified constraint is verified by $c'$ then
 6:       add $c'$ in $L$ as successor of $c$ if $c'$ does not exit in $L$ else connect $c'$ as
          successor of $c$.
 7:     end if
 8:   end for
 9: until no concept is added in $L$.
10: derive the neural network architecture as described in section 3.3 from the concept
    semi-lattice.
```

3.2 Constraints

In order to reduce the size of lattice and then the time complexity, we present a few constraints regularly used to select concepts during the learning process.

Frequency of concept. A concept is frequent if it contains at least α (also called minsupp is specified by the user) objects. The support s of a concept (X, Y) is the ratio between the cardinality of the set X and the total number of objects $(|O|)$ $(s = \frac{100 \times |X|}{|O|}\%)$. Frequency is an anti-monotone constraint which helps in pruning the lattice and reduce it computational complexity. Support could be seen as the minimal number of objects that the intention of one concept must verified before being taken in the semi-lattice. The figure 3.2 presents a semi-lattice built from the class "+" examples of the table 1 (a) and the equivalent topology of the neural network (b); the specified minsupp value is 30%.

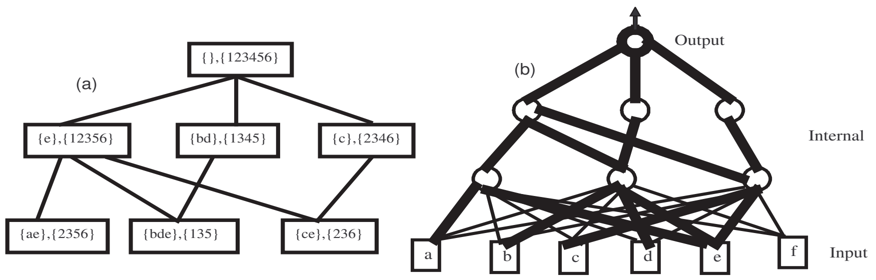


Fig. 2. (a) Join semi-lattice with minsup=30% and (b) network topology.

Validity of concept. Many techniques are used to reduce the size of lattice. The following notions are used in order to the select concepts: a concept (X, Y) is **complete** if Y recognize all examples in dataset. A concept (X, Y) is **consistent** if Y throws back all counterexamples (formally, the set of consistent concept is $\{(X, Y)/Y \cap O^- = \{\}\}$ where $O = O^+ \cup O^-$). To reduce the restriction imposed by these two constrains, others notions are used:

1. **Validity**. A concept (X, Y) is valid if its description recognizes most examples; a valid concept is a frequent concept on the set of examples O^+; formally the set of valid concepts is defined as $\{(X, Y) \,/\, |X^+| \geq \alpha\}$ where $0 < \alpha \leq |O^+|$.
2. **Quasi-consistency**. A concept (X, Y) is quasi-consistent is if it is valid and its extension contains few counterexamples. Formally the set of quasi-consistent concepts is defined as $\{(X, Y) \,/\, |X^+| \geq \alpha \textit{ and } |X^-| \leq \beta\}$.

Height of semi-lattice. The level of a concept c is defined as the minimal number of connexions from the supremum concept to c. The height of the lattice is the greatest value of the level of concepts. Using levelwise approach to generate the join semi-lattice, a given constraint can be set to stop concept generation at a fixed level. The height of the lattice could be performed as the depth without considering the cardinality of concepts extension (or intention). In fact at each level, concept extensions (or intentions) do not have the same cardinality. The number of layers of the semi-lattice is a parameter corresponding to the maximum level (height) of the semi-lattice.

3.3 From Semi-Lattice to Neural network

Mapping Hasse diagram of lattice into neural networks is described as following:

– Each lattice concept becomes one unit (or node) in the neural network; a node is thus seen as a group of objects which verify a given set of attributes.
– A layer of n units (n is the number of attributes) is created as the input layer of the system; each neuron of this layer is connected to any neuron of the first hidden layer (concepts in the semi-lattice, with no successor) inside internal layers.
– The supremum concept of lattice is translated into the neural unit representing the output of the network; this concept is the one whose extension contains all the training set of objects.
– Other concepts are rewritten as hidden units (or hidden layers). The units which do not have successor constitute the first hidden layer.
– There is a connection between two neurons if and only if there is a link between their associated concepts in the join semi-lattice.

3.4 Connection weights and threshold

Connection weights are initialized as follows:

– Connection between two nodes, that derived directly from the semi-lattice is initially weighted to 1. This implies that when the node (or neuron) is active, all its predecessors are active too.
– Connection between an input layer unit and hidden layer unit is weighted as follows: 1 if the attribute associated by the input layer node appears in the intention Y of concept associated to the hidden layer node, and -1 otherwise.
– Threshold is set to zero for all nodes (or units).

These connection weights are modified during the training process, using the learning algorithm (which is the backpropagation algorithm [2] in our case). The activation function is also a parameter of CLANN approach. We choose to implement the sigmoid function among those reported in the literature [2].

4 EXPERIMENTATIONS AND RESULTS

This section presents the experimental comparisons conducted with our method. CLANN has been implemented in C++, and run on a personal computer Pentium IV (1,8Ghz, 1Go RAM and 80Go Hard Disk) in the Linux fedora core environment. Table 2 describes the two-class datasets with contain nominal attributes, taken from UCI repository [3]. We use the Weka binarization procedure "NominalToBinary" to convert multivalued attribute to binary one.

Dataset	#Nominal attributes	#Binary attributes	training	test	data-size
Spect	23	23	80	187	267
Chess end-of-game	36	38	10-CV	10-CV	3196
Tic-tac-toe	9	27	10-CV	10-CV	958
Monsk1	7	15	124	432	556
Monsk2	7	15	168	432	600
Monsk3	7	15	122	432	554

Table 2. Experimented Datasets; '10-CV' indicates that training and test data are defined by 10-fold cross validation technique

Neural network obtained from the Hasse diagram of the join semi-lattice are trained with backpropagation algorithm using 500 iterations. Activation function used in the experiments is the sigmoid function ($f(x) = \frac{1}{1+\exp(x)}$). The experimentations were done using 10-fold cross validation for datasets with none already defined test set as Chess and Tic-tac-toe. For other datasets, our model is built and trained using provided training dataset, and evaluate on test set. CLANN results are depicted by three tables : 3, 4 and 5. These tables present the accuracy rate which is the percentage of correctly classified test objects. Table 3 shows the results obtained on one dataset (SPECT) with the different constraints and various parameter values. Considering the frequency or validity

[3] available on the web site http://www.ics.uci.edu/AI/ML/MLDBRepository.html

constraint, when the minimum support (δ or α) decreases, the accuracy rate increases. This is certainly due to the fact that the number of hidden nodes also increases when the minimum support decreases. With a high number of hidden nodes, the neural network model produces good perfomance. Considering the quasi-consistency constraint, when its minimum value decreases, the accuracy rate slightly decreases. The effect of this constraint is not so significant in our case. Considering the semi-lattice height constraint, the accuracy rate slightly increases until level 2, and then significantly decreases. Thus it is not necessary to build more than two levels from of the semi-lattice. We choose to use this constraint (setting the default value to 1) in order to compare CLANN to other machine learning classifiers.

Table 3. Results obtained from SPECT database using different constraints.

Constraints	Generalization	Constraints	Generalization
α=10, β=10	93,74%	δ=20	93,59%
α=20, β=10	92,60%	δ=30	93,59%
α=30, β=10	91,30%	δ=40	90,65%
α=40, β=10	90,4%	δ=50	89,84%
α=10, β=50	93,90%	*height*=1	93,60%
α=20, β=50	93,90%	*height*=2	93,90%
α=30, β=50	92,40%	*height*=3	92,40%
α=40, β=50	88,40%	*height*=4	90,65%

Table 4 presents accuracy rates of CLANN and four other classifiers taken from the Weka datamining tool [4] among which two decision tree method (C4.5 and ID3), one instance based learning method (IB1), and one neural network method (MLP). MLP is built with the ad-hoc method as described in the section 1 and trained using gradient backpropagation algorithm [12]. Considering the accuracy rate average over the datasets used, CLANN is never the last classifier when ranking them on the accuracy rate. It is always better than at least one other classifier. MLP outperforms all the other classifier in terms of accuracy rates, however it is not a comprehensible model for datamining. CLANN results are comparable to decision-tree ones, and CLANN outperforms IB1. Considering the SPECT dataset, CLANN significantly outperform all those classifiers.

Table 5 presents comparative results between CLANN and other constructive neural networks models [11] among which Tiling, Upstart, Tower, Distal. Those constructive neural networks are still non compresensible model for data mining. Our comparison is only to analyse the soundness of our approach. The three first algorithms construct the neural model by successive addition of neurons until the network has a maximum number (specified by the user) of layers or the desired accuracy is obtained. The parameters used are described as following: the maximum number of layer is 10 and the desired accuracy is 100% (default value

[4] available on the web site http://www.cs.waikato.ac.nz/ml/weka/

Table 4. Accuracy rate of CLANN versus other standard models

Dataset	CLANN	MLP	C4.5	ID3	IB1
Spect	93,90%	65,77%	66,7%	65,24%	66,31%
Chess	93,60 %	99,30%	98,30%	97,80%	89,90%
Monsk1	82,70%	100%	100%	92,59%	89,35%
Monsk2	78,91%	100%	70,37%	86,57%	66,89%
Monsk3	83,61%	93,52%	100%	89,81%	81,63%
Tic-tac-toe	83,57%	96,86%	93,21%	93,84%	81,63%
Average	86,05%	92,56%	88,10%	87,64%	79,29%

used in their implementation), added unit is trained by Pocket Perceptron with racket modification [11]. Distal uses one hidden layer and clusters the training set into disjoint groups and each group becomes one neural unit of the hidden layer. Distal is similar to our approach. They differ by the fact that: Distal uses only one hidden layer, Distal uses disjoint group while the node extension in CLANN may overlap and furthermore each node in CLANN model is clearly characterized by an intention.

Table 5. Classification rate of CLANN versus other constructive neural network models

Dataset	CLANN	Tiling	Upstart	Tower	Distal
Spect	93,90%	89,60%	83,29%	71,40%	93,90%
Chess	93,60%	93,90%	90,65	92,40%	89,74%
Monsk1	82,70%	83,13%	77,78%	85,85%	90,23%
Monsk2	78,91%	77,13%	87,30%	66,90%	89,10%
Monsk3	83,61%	74,91%	87,22%	82,08%	86,46%
Tic-tac-toe	83,57%	98,40%	99,89%	100%	95,85%
Average	86,05%	86,17%	87,67%	83,11%	90,88%

From table 5, considering the accuracy rate average, Distal outperforms all the classifiers, CLANN outperforms Tower and is comparable to Tiling and Upstart. As shown in the previous comparison, CLANN is better on the SPECT dataset. CLANN outperforms Distal on CHESS dataset. In addition of these results, CLANN built a comprehensive model and is more suitable for data mining than the other constructive approaches. In fact concept lattices can be useful in rules extraction process [6]. The other constructive neural network model as well as MLP are seen as black box by the user, and their interpretation remains a difficult task. Providing explanation when dealing with datamining applications brings up an added-value [1, 3], which will be easily obtained with CLANN than other methods.

KBANN [13] is not compared to CLANN because the apriori knowledge about the previous dataset (table 2) are not available.

5 RELATED WORKS

To the best of our knowledge, KBANN [13] is the only algorithm able to build comprehensive neural network. KBANN derives from the knowledge provided by the user as set of rules, the network topology. Unlike KBANN, CLANN does not need apriori knowledge, its model is based on semi-lattice built directly from data.

Research works about rules extraction from neural networks presented two main algorithms for this task. These algorithms [1] execute on neural model according to the assumptions which stipulate that neurons are maximally active (activation near 1 or 0) and that training does not significantly alter the meaning of the neuron, implicite rules present in the model after mapping Hasse diagram into neural network will not significantly change after training. These algorithms are:

1. Subset. Its principle [1] consists of clustering internal links of hidden and output units, into subsets of links, and then using only the subsets of links whose the sum of weights is greater than the unit threshold. Extracted rules are in written in the form ”If...then...”
2. MofN. It assumes that individual links are not important and weights form small number of clusters. It clusters the connections for each hidden and output neuron and delete those clusters which can not affect the unit result. It finds the average weight for each remaining cluster and retrain the bias of each cluster. Finally a single rule for each hidden and output neuron X is derived. The rules are in the form ”If N of the M antecedents are active then X is active”.

Each of these algorithms could be applied to extract rules in CLANN model.

As the architecture of CLANN model is based on semi-lattice, extracting rules using concept lattices approaches [9] could also be applied. Each node of CLANN model contains a closed set and the connexion weight between two nodes could help to weight the extracted rules.

6 CONCLUSION

This work proposes an approach to define a constructive comprehensive neural network without an apriori knowledge. Experiments conducted has shown the soundness and efficiency of our approach.

CLANN is limited to the binary data set. Binarization techniques could be applied on dataset such that CLANN is able to handle nominal, discrete and continuous values. Various approaches are proposed in the literature to deal with many-valued context directly when generating concept lattice. Further research will focus on those approaches. Many constraints are also presented but it is not easy to define the default values. Extensive experimental study is currently ongoing to find out the default values. Computational complexity of CLANN is similar to those of Weka MLP, this complexity will be further explicitly explored.

CLANN is also limited to two-class supervised classification problem. Ongoing research is dealing with multi-class design, constraint criteria setting and rules extraction from CLANN topology.

Acknowledgment

This research is partially supported by the French Embassy service SCAC (Service de la Coopération et d'Action Culturelle) in Yaounde. Thanks to the anonymous reviewers for their helpful comments.

References

1. R. Andrews, J. Diederich, A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-Based Systems 8(6): 373-389, 1995.
2. Y. Benani. Apprentissage Connexionniste, Editions Herms Science, Paris 2006.
3. M.W. Craven and J.W. Shavlik. Using Neural Networks for Data Mining. Future Generation Computer Systems, 13(2-3): 211-229, 1997.
4. S. Gallant. Perceptron based learning algorithms. IEEE Transactions Neural Networks, 1:179-191, 1990.
5. B. Ganter and R.Wille. Formal Concepts Analysis: Mathematical foundations. Springer - Verlag,1999.
6. G. Gasmi, S. Ben Yahia, E. Mephu Nguifo, Y. Slimani. A new informative generic base of association rules. Advances in Knowledge Discovery and Data Mining, 3518:81-90,2005.
7. J. Han, M. Hamber. Datamining: Concepts and Techniques. Morgan Kauffman Publishers, London, 2001.
8. S. Kuznetsov, S. Obiedkov. Comparing Performance of Algorithms for Generating Concept Lattices, JETAI 14(2/3):189-216,2002.
9. J.L Guigues and Vincent Duquenne, 'Familles minimales d'implications informatives resultant d'un tableau de donnes binaires, Mathmatiques et sciences sociales, 95:5-18 1986.
10. E. Mephu Nguifo, P. Njiwoua. Treillis de concepts et classification supervisée. Techniques et Sciences Informatiques, 24:449-488, 2005.
11. R. Parekh, J. Yang, V. Honavar. Constructive Neural-Network Learning Algorithms for Pattern Classification. IEEE Transactions on Neural Networks, 11(2): 436-451, 2000.
12. D.E. Rumelhart, G. E. Hinton, R. J. Williams. Learning representations by backpropagating errors. Nature,323: 318-362, 1986.
13. J.W. Shavlik, G. Towell. Kbann: Knowledge based articial neural networks, Artificial Intelligence, 70:119-165, 1994.

A Proposal of Description Logic on Concept Lattices*

N.V. Shilov[1,2] and S.-Y. Han[1]

[1] School of Computer Science and Engineering
Chung-Ang University 410, 2nd Engineering Building 221,
Heukseok-dong, Dongjak-gu Seoul 156-756, Korea
shilov@cau.ac.kr, hansy@cau.ac.kr
[2] A.P. Ershov Institute of Informatics Systems,
Lavren'ev av., 6, Novosibirsk 630090, Russia,
shilov@iis.nsk.su

Abstract. There are two major formalisms that are developed around concepts: (1) Formal Concept Analysis (FCA) by R. Wille and B. Ganter, and (2) Description Logic (DL) that goes back to the universal terminological logic by P.F. Patel-Schneider. It has been demonstrated that FCA constructs (upper and lower derivatives, formal concepts) are expressible in DL. Present paper demonstrates how to interpret (positive) DL concepts over concept lattices in a compatible manner.

1 Introduction

Two major formalisms developed around concepts are

– Formal Concept Analysis (FCA) by R. Wille and B. Ganter [4, 12],
– Description Logics (DL) that go back to P.F. Patel-Schneider [2].

Description Logic (DL) is a set of knowledge representing languages closely related to modal [3] and program logics [6]. These languages can be used for description of the terminological knowledge in a structured way. They have become a cornerstone of the W3C-endorsed Web Ontology Language (OWL) [7]. DL basic notions are concept and role terms. Concept terms correspond to formulas of modal and program logics. An interpretation assigns sets of objects (that are called concepts in DL) to concept terms. The most important DL notion is knowledge base. A knowledge base is a collection of (subsumption) statements between pairs of concept terms and pairs of role terms. In this way DL represents that a concept is a subconcept of another one, a role is a subrole of another role.

* This research is supported in parts by (a) MIC(Ministry of Information and Communication), Korea, under the Foreign Professor Invitation Program of the IITA(Institute for Information Technology Advancement), (b) by joint grant RFBR 05-01-04003-a - DFG project COMO, GZ: 436 RUS 113/829/0-1, (c) by Integration Grant n.14 Siberia Branch, Russian Academy of Science.

Formal concept analysis is an algebraic framework for data representation and analysis. It takes an input matrix that specifies a relation between objects and attributes. Then FCA finds corresponding 'closed' sets of attributes and 'closed' sets of objects. Every pair of corresponding 'closed' sets of objects and attributes forms a formal concept[3]. The set of attributes in each formal concept can be interpreted as a set of necessary and sufficient conditions for defining the set of objects in the concept. The family of formal concepts obeys the mathematical axioms defining a lattice, and is called a concept lattice.

To the best of our knowledge, there are few research on combination of Formal Concept Analysis with Description Logic [9, 1, 8]. Roughly speaking, all these attempts can be classified as follows:

– accelerating one formalism by another,
– emulating one formalism by another.

The dissertation [9] and paper [1] attempted to accelerate DL by variants of a so-called attribute exploration technique that goes back to FCA. A variant suggested by [9] is called Relational Exploration. It allows determining all 'attribute implications' that follows from knowledge base presented in terms of a simple description logic FLE. (This logic admits concept intersections, existential and universal restrictions.) A variant of attribute exploration proposed by [1] can deal with a partial context and any description logic that is closed under complement and conjunction. An algorithm developed in the cited paper constructs a complete extension of a given partial knowledge base and at the same time it guarantees a minimum communication with knowledge engineers.

In contrast, paper [8] has attempted to emulate DL universal restriction in FCA by some algebraic fix-point construction and to demonstrate its utility for analysis of relational data.

The present paper discusses two variants of a 'combination' of DL and FCA. The first variant is sketched in brief since it has been discussed in full details in a recent workshop paper [11]. This variant emulates FCA in DL by extending language of any description logic $\mathcal{L}$ by upper and lower derivatives that come from FCA. The resulting description logic is denoted by $\mathcal{L}$/FCA. Paper [11] has proved that $\mathcal{L}$/FCA can be expressed in $\mathcal{L}(\neg, -)$ – another variant of $\mathcal{L}$ that is closed with respect to role complement and inverse.

A new variant of a 'combination' of DL and FCA emulates a particular description logic – (positive fragment of) Attribute Language with Complements ($\mathcal{ALC}$) [10] – by algebraic operations on concept lattices. This emulation can be considered also as a usage of concept lattices as domains for terminological interpretations, i.e. as a conceptual semantics for $\mathcal{ALC}$. Informal soundness of this emulation is supported by a compatibility of the conceptual semantics with the standard semantics of $\mathcal{ALC}$.

The paper is organized as follows. The next section introduces basic notions of Description Logic. Then Section 3 presents basic definitions and some facts of Formal Concept Analysis. In Section 4 we present our proposal of a description

[3] For explicit distinction with DL, we use here a combined term 'formal concept'.

logic on concept lattices. Section 5 discuses the compatibility of the conceptual semantics of $\mathcal{ALC}$ with the standard one. Topics for further research are discussed in Conclusion.

2 Basics of Description Logics

There are many variants of description logics, but we define only some of them.

Definition 1. *Syntax of every description logic is constructed from disjoint alphabets of concept, role, and object symbols CS, RS, and OS, respectively. The sets of concept terms (or concepts) CT and role terms (or roles) RT are defined by induction. Usually definition admits the following clauses.*

– *(Concept terms)*
  • *the top concept $\top$ and the bottom concept $\bot$ are concept terms;*
  • *any concept symbol is a concept term;*
  • *for any concepts X and Y their union $(X \sqcup Y)$ and intersection $(X \sqcap Y)$ are concept terms;*
  • *for any concept X its complement $(\neg X)$ is a concept term;*
  • *for any role R and any concept X the universal $(\forall R.\ X)$ and the existential $(\exists R.\ X)$ restrictions are concept terms;*

– *(Role terms)*
  • *the top role ∇ and the bottom role $\triangle$ are role terms;*
  • *any role symbol is a role term;*
  • *for any roles R and S their union $(R \sqcup S)$, intersection $(R \sqcap S)$, and composition $(R \circ S)$ are terms;*
  • *for any role R its complement $(\neg R)$, inverse (R^-), and transitive closure (R^+) are role terms.*

Concept and role terms altogether form the set of terminological expressions.

Definition 2.

– *For any concepts X and Y, any roles R and S the following expressions are called terminological sentences: $X \dot{\sqsubseteq} Y$, $X \doteq Y$, $R \dot{\sqsubseteq} S$, and $R \doteq S$. A TBox is a set of terminological sentences.*
– *For any concept X, any role R, and any object symbols a and b the following expressions are called assertional sentences: a concept assertion $a : X$ and a role assertion $(a, b) : R$. An ABox is a set of assertional sentences.*
– *A knowledge base is a finite set of terminological and assertional sentences. Every knowledge base consists of an appropriate TBox and ABox.*

Definition 3. *Semantics of any description logic is defined in Kripke-like terminological interpretations. Every terminological interpretation is a pair (D, I), where D is a set (that is called domain) and I is a mapping (that is called interpretation function). This function maps object symbols to elements of D, concept symbols to subsets of D, role symbols to binary relations on D: $I =$*

$I_{OS} \cup I_{CS} \cup I_{RS}$, where $I_{OS} : OS \to D$, $I_{CS} : CS \to 2^D$, and $I_{RS} : RS \to 2^{D\times D}$. The unique name assumption holds for this function: $I(a) \neq I(b)$ for all different object symbols a and b. The interpretation function can be extended to all terminological expressions as follows.

- *(Concept semantics)*
  - $I(\top) = D$ *and* $I(\bot) = \emptyset$;
  - $I(X \sqcup Y)) = I(X) \cup I(Y)$ *and* $I(X \sqcap Y) = I(X) \cap I(Y)$;
  - $I(\neg X) = D \setminus I(X)$;
  - $I(\forall R.\ X) = \{s \in D : \forall t \in D(\mathit{if}\ (s,t) \in I(R)\ \mathit{then}\ t \in I(X))\}$,
    $I(\exists R.\ X) = \{s \in D : \exists t \in D((s,t) \in I(R)\ \mathit{and}\ t \in I(X))\}$;
- *(Role semantics)*
  - $I(\nabla) = D^2$ *and* $I(\triangle) = \emptyset$;
  - $I(R \sqcup S) = I(R) \cup I(S)$, $I(R \sqcap S) = I(R) \cap I(S)$, $I(R \circ S) = I(R) \circ I(S)$ *(righthand side '$\circ$' is composition of binary relations);*
  - $I(\neg R) = D^2 \setminus I(R)$, $I(R^-) = (I(R))^-$, *and* $I(R^+) = (I(R))^+$ *(righthand side '$^+$' is transitive closure of binary relations);*
  - $I(R!X) = \{(s,t) \in I(R) : s \in I(X)\}$,
    $I(R?X) = \{(s,t) \in I(R) : t \in I(X)\}$.

Definition 4. *Semantics of sentences is defined in terminological interpretations in terms of satisfiability relation as follows:*

- $(D,I) \models a : X$ *iff* $I(a) \in I(X)$;
- $(D,I) \models X \dot{\sqsubseteq} Y$ *iff* $I(X) \subseteq I(Y)$;
- $(D,I) \models X \doteq Y$ *iff* $I(X) = I(Y)$;
- $(D,I) \models (a,b) : R$ *iff* $(I(a), I(b)) \in I(R)$;
- $(D,I) \models R \dot{\sqsubseteq} S$ *iff* $I(R) \subseteq I(S)$;
- $(D,I) \models R \doteq S$ *iff* $I(R) = I(S)$.

This satisfiability relation can be extended on knowledge bases in a natural way: $(D,I) \models KBase$ iff $(D,I) \models \phi$ for every sentence $\phi \in KBase$. In the case of $(D,I) \models KBase$, the terminological interpretation (D,I) is said to be a (terminological) model for the knowledge base $KBase$. Let us say that a knowledge base $KBase$ entails a sentence ψ (and write[4] $Kbase \models \psi$) iff $(D,I) \models \psi$ for every model (D,I) of $KBase$.

Definition 5. *A concept X is said to be coherent (or satisfiable) with respect to a knowledge base $KBase$ iff there exists a terminological model (D,I) for $KBase$ such that $I(X)$ is not empty. A knowledge base $KBase$ is said to be satisfiable iff the top concept $\top$ is coherent with respect to $KBase$.*

Attribute Language with Complements ($\mathcal{ALC}$) [10] is a particular example of description logic. In simple words, $\mathcal{ALC}$ adopts role symbols as the only role terms, concept symbols – as elementary concept terms, and permits 'Boolean' constructs '$\neg$', '$\sqcup$', '$\sqcap$', universal and existential restrictions '$\forall$' and '$\exists$' as the only concept constructs. The formal definition follows.

[4] When $KBase$ is empty then '$\models \psi$' can be written instead of '$\emptyset \models \psi$'.

Definition 6. *$\mathcal{ALC}$ is a fragment of DL that comprises concepts that are defined by the following context-free grammar:*

$$C_{\mathcal{ALC}} ::= \\ CS|\top|\bot|(\neg C_{\mathcal{ALC}})|(C_{\mathcal{ALC}} \sqcup C_{\mathcal{ALC}})|(C_{\mathcal{ALC}} \sqcap C_{\mathcal{ALC}})|(\forall RS.\ C_{\mathcal{ALC}})|(\exists RS.\ C_{\mathcal{ALC}})$$

where metavariables CS and RS represent any concept and role symbols, respectively. Semantics of $\mathcal{ALC}$ is defined in the standard way in accordance with Definition 3.

Many description logics can be defined as extensions of $\mathcal{ALC}$ by concept and/or role constructs. For example, the website [13] uses the following approach: for any collection of concept and/or role constructs $C\&R$, let $\mathcal{ALC}(C\&R)$ be a closure of $\mathcal{ALC}$ that admits all concept and/or role constructs in $C\&R$.

Definition 7. *A variant of DL is a description logic $\mathcal{L}$ with syntax that*

- *contains all concept and role symbols CS and RS,*
- *is closed under concept constructs '$\neg$', '$\sqcup$', '$\sqcap$', '$\forall$' and '$\exists$'.*

From the viewpoint of the above definition, $\mathcal{ALC}$ is the smallest variant[5] of DL.

Definition 8. *Let $\mathcal{L}$ be a variant of DL and $C\&R$ be a collection of concept and/or role constructs. Then let $\mathcal{L}(C\&R)$ be the smallest variant of DL that includes $\mathcal{L}$ and is closed under all constructs in $C\&R$.*

For instance, $\mathcal{ALC}(\neg, -)$ is an extension of $\mathcal{ALC}$ where any role symbol can be negated and/or inverted.

The following definition takes into account DL variants without role constructs for so-called domain restriction and range restriction.

Definition 9. *A concept term is said to be positive if it does not use the concept complement construct. For every DL variant $\mathcal{L}$ the positive fragment of $\mathcal{L}$ comprises all positive concept terms of $\mathcal{L}$ and all role terms of $\mathcal{L}$. The positive fragment of DL variant $\mathcal{L}$ is denoted by $\mathcal{L}^+$.*

For example $\mathcal{ALC}^+$, the positive fragment of $\mathcal{ALC}$, comprises concepts that are defined by the following context-free grammar:

$$C_{\mathcal{ALC}^+} ::= \\ CS|\top|\bot|(C_{\mathcal{ALC}^+} \sqcup C_{\mathcal{ALC}^+})|(C_{\mathcal{ALC}^+} \sqcap C_{\mathcal{ALC}^+})|(\forall RS.\ C_{\mathcal{ALC}^+})|(\exists RS.\ C_{\mathcal{ALC}^+}).$$

The following lattice-theoretic characterization of semantics of concept constructs '$\top$', '$\bot$', '$\sqcup$', '$\sqcap$', '$\forall$', and '$\exists$' is easy to prove.

Proposition 1. *Let (D, I) be a terminological interpretation and $P(D) = (2^D, \emptyset, \subseteq, D, \cup, \cap)$ be the complete lattice of subsets of D. Then $\mathcal{ALC}^+$ semantics enjoys the following properties in $P(D)$:*

[5] Of course, 'smaller' description logics can be defined by means of stronger syntax restrictions.

– *for any concept symbol $I(p)$ is an element of the lattice $P(D)$;*
– $I(\top) = \sup P(D)$ *and* $I(\bot) = \inf P(D)$;
– $I(X \sqcup Y) = \sup(I(X), I(Y))$, *and* $I(X \sqcap Y) = \inf(I(X), I(Y))$;
– $I(\forall R.\ X) = \sup\{S \in P(D) : \forall s \in S \forall t \in D((s,t) \in I(R) \Rightarrow t \in I(X))\}$,
  $I(\exists R.\ X) = \sup\{S \in P(D) : \forall s \in S \exists t \in D((s,t) \in I(R)\ \&\ t \in I(X))\}$.

3 Basics of Formal Concept Analysis

Basic Formal Concept Analysis (FCA) definitions below follow monograph [4], but we use a little bit different notation.

Definition 10. *A formal context is a triple (O, A, B) where O and A are sets of 'objects' and 'attributes' respectively, and $B \subseteq O \times A$ is a binary relation connecting objects and attributes. Let us say that a formal context (O, A, B) is homogeneous*[6] *iff $O = A$, i.e. the set of objects coincides with the set of attributes. A symmetric context is a homogeneous context with symmetric binary relation*[7]

Every terminological interpretation (D, I) and every role term R define a formal context $(D, D, I(R))$. It implies that every terminological interpretation (D, I) defines a family of homogeneous formal contexts $(D, D, I(R))$ indexed by role symbols $R \in RS$ or by role terms $R \in RT$.

Vise versa, there is a number of ways to define a terminological interpretation for given formal contexts. For example, if we have a family of formal contexts (O_j, A_j, B_j) indexed by elements of some set J, then we can adopt the set of indexes J as the alphabet role symbols RS, a set of symbols $\{o_j : j \in J\} \cup \{a_j : j \in J\}$ as the alphabet of concept symbols CS, and define a terminological interpretation (D, I) as follows:

– $D = \cup_{j \in J}(O_j \cup A_j)$,
– $I(j) = B_j \subseteq (O_j \times A_j) \subseteq D \times D$ for every $j \in J$,
– $I(a_j) = A_j \subseteq D$ and $I(o_j) = O_j \subseteq D$ for every $j \in J$.

Two major algebraic operations for formal contexts are upper and lower derivations. These operations are used in the definition of a notion of a formal concept, its extent and intent.

Definition 11. *Let (O, A, B) be a formal context.*

– *For every set of objects $X \subseteq O$ its upper derivation $X^{\uparrow}$ is the following set of attributes $\{t \in A :$ for every $s \in O$, if $s \in X$ then $(s,t) \in B\}$, i.e. the collection of all attributes that all objects in X have simultaneously.*
– *For every set of attributes $Y \subseteq A$ its lower derivation $Y^{\downarrow}$ is the following set of objects $\{s \in O :$ for every $t \in A$, if $t \in Y$ then $(s,t) \in B\}$, i.e. the collection of all objects that all attributes in Y have simultaneously.*

[6] 'Homogeneous context' is our own term.
[7] Recall that a binary relation B is symmetric, if $(x, y) \in B \Leftrightarrow (y, x) \in B$ for all x and y. 'Symmetric context' is also our own term.

- *A formal concept is a pair (Ex, In) such that $Ex \subseteq O$, $In \subseteq A$, and $Ex^{\uparrow} = In$, $In^{\downarrow} = Ex$; components Ex and In of the formal concept (Ex, In) are called its extent and its intent respectively.*

Definition 12. *For every formal context $K = (O, A, B)$*

- *let $FC(K)$ be the set of all formal concepts over K, $\top_K$ be a formal concept $(O, O^{\uparrow})$, and $\bot_K$ be a formal concept $(A^{\downarrow}, A)$;*
- *let $\preceq_K$ be the following binary relation $FC(K)$:*
  $(Ex', In') \preceq_K (Ex'', In'')$ *iff* $Ex' \subseteq Ex''$ *and/or*[8] $In'' \subseteq In'$;
- *let $\sup_K$ be the following operation on subsets of $FC(K)$:*
  $\sup_K\{(Ex_j, In_j) \in FC(K) : j \in J\} \;=\; ((\cup_{j\in J} Ex_j)^{\uparrow\downarrow}\;,\; \cap_{j\in J} In_j)$;
- *let $\inf_K$ be the following operation on subsets of $FC(K)$:*
  $\inf_K\{(Ex_j, In_j) \in FC(K) : j \in J\} \;=\; (\cap_{j\in J} Ex_j\;,\; (\cup_{j\in J} In_j)^{\downarrow\uparrow})$.

The following fact is a part of the Basic Theorem on Concept Lattices [4] (Theorem 3).

Fact 1 *For any formal context K an algebraic system $(FC(K), \preceq_K, \top_K, \bot_K, \sup_K, \inf_K)$ is a complete lattice.*

The definition below makes sense due to the above theorem.

Definition 13. *For every formal context K let the concept lattice $CL(K)$ be a complete lattice $(FC(K), \preceq_K, \top_K, \bot_K, \sup_K, \inf_K)$.*

The next fact is a corollary of the Basic Theorem on Concept Lattices.

Fact 2 *Every complete lattice is isomorphic to some concept lattice.*

For example, for any set D the complete lattice $P(D) = (2^D, \emptyset, \subseteq, D, \cup, \cap)$ is isomorphic to the concept lattice of a homogeneous context $K^{\neq}_D = (D, D, \neq)$. A particular isomorphism is a function $\iota : 2^D \to FC(K^{\neq}_D)$ that maps every subset $S \subseteq D$ to a formal concept $(S, \overline{S})$. Let us remark that for every $S \subseteq D$, $\iota(\overline{S}) = (\overline{S}, S)$, i.e. permutation of $\iota(S)$. – Let us refer this observation by complement property of the powerset concept lattice.

4 Towards Description Logics on Concept Lattices

One can observe that 'concepts' in Description Logic and in Formal Concept Analysis are of different nature. The former are just subsets of domains in terminological interpretations, the later are compatible pairs of object and attribute sets in formal contexts.

We see two opportunities to combine these two formalizations of 'concepts'. The first opportunity is 'integration' of basic constructions of Formal Concept Analysis to a framework of Description Logic. The second opportunity is to 'interpret' concept terms by formal concepts.

[8] These two conditions are equivalent each other.

The first opportunity has been examined in a workshop paper [11]. But there was criticism after publication of [11], that the paper has adopted a pure set-theoretic approach to concepts. As a consequence of this, a lattice-theoretic structure (that is very special advantage of Formal Concept Analysis) has been lost. Hence it is important to investigate the second opportunity of combination of FCA and DL and develop (in a compatible manner) a description logic directly on concept lattices. Below we present our proposal of a particular variant of description logic on concept lattices.

Syntax of description logics on concept lattices is the same as in Definition 1. Semantics of description logics on concept lattices comes from lattice-theoretic characterization of semantics of 'positive' concept constructs that is given in Proposition 1 and the complement property of the powerset concept lattice. Conceptual interpretation is a formal context provided by an interpretation function.

Definition 14. *Conceptual interpretation is a four-tuple (O, A, B, I) where (O, A, B) is a context, and I is an interpretation function. This function maps object symbols to elements of O, attribute symbols to elements of A, concept symbols to formal concepts in $FC(O, A, B)$, role symbols to binary relations on $O \cup A$: $I = I_{OS} \cup I_{AS} \cup I_{CS} \cup I_{RS}$, where $I_{OS} : OS \to O$, $I_{AS} : AS \to A$, $I_{CS} : CS \to CL(O, A, B)$, and $I_{RS} : RS \to 2^{(O\times O)\cup(A\times A)}$. The unique name assumption holds for this function: $I(o') \neq I(o'')$ and $I(a') \neq I(a'')$ for different object and attribute symbols o', o'', a', and a''. A conceptual interpretation (O, A, B, I) is said to be homogeneous (symmetric), if (O, A, B) is a homogeneous (symmetric respectively) context.*

Definition 15. *Semantics of any description logic on concept lattices is defined by means of conceptual interpretations. Let (O, A, B, I) be a conceptual interpretation, K be a formal context (O, A, B), and $CL(K)$ be a concept lattice $(FC(K), \preceq_K, \top_K, \bot_K, \sup_K, \inf_K)$. The interpretation function I can be extended to all role terms in a terminological interpretation $((O \cup A), I)$ in the standard manner in lines with the definition 3 so that $I(R)$ is a binary relation on $(O \cup A)$ for every role term R. The interpretation function I can be extended to all concept terms as follows.*

- *$I(\top) = \sup_K FC(K)$ and $I(\bot) = \inf_K FC(K)$;*
- *$I(X \sqcup Y) = \sup_K(I(X), I(Y))$, and $I(X \sqcap Y) = \inf_K(I(X), I(Y))$;*
- *Let K be a symmetric context and $I(X) = (Ex, In) \in CL(K)$. Then $I(\neg X) = (In, Ex)$.*
- *Let $I(X) = (Ex', In') \in CL(K)$. Then*
  - *$I(\forall R.\ X) = \sup_K\{(Ex, In) \in CL(K) :$ $\forall o \in Ex\ \forall a \in In\ \forall o' \in O\ \exists a' \in A$ $((o, o') \in I(R) \Rightarrow o' \in Ex',\ (a, a') \in I(R),$ and $a' \in In')\}$,*
  - *$I(\exists R.\ X) = \sup_K\{(Ex, In) \in CL(K) :$ $\forall o \in Ex\ \forall a \in In\ \exists o' \in O\ \forall a' \in A$ $((a, a') \in I(R) \Rightarrow (o, o') \in I(R),\ o' \in Ex',$ and $a' \in In')\}$.*

Proposition 2.

1. *For any conceptual interpretation (O, A, B, I), for every positive concept term X, semantics $I(X)$ is an element of $FC(O, A, B)$.*
2. *For any symmetric conceptual interpretation (D, D, B, I), for every concept term X, semantics $I(X)$ is an element of $FC(D, D, B)$.*

Proof. By induction on structure of a (positive) concept. It is trivial in all cases but exploits (a) completeness of concept lattices (i.e. sup always exists), and (b) that permutation of extent and intent of a formal concept is also a formal context in every symmetric context. ■

It follows from the above proposition that in general the positive fragment $\mathcal{ALC}^+$ is 'the smallest' description logic on concept lattices, while $\mathcal{ALC}$ is 'the smallest' description logic on symmetric concept lattices.

5 $\mathcal{ALC}$ on a powerset concept lattice

Let (D, I) be a terminological interpretation. We remarked in section 3 (after fact 2), that the powerset lattice $P(D) = (2^D, \subseteq, \emptyset, D, \cup, \cap)$ is isomorphic to the concept lattice of a homogeneous formal context $K^{\neq}_D = (D, D, \neq)$. A particular isomorphism is a function $\iota : 2^D \to FC(K^{\neq}_D)$ that maps every subset $S \subseteq D$ to a formal concept $(S, \overline{S})$. This isomorphism defines conceptual interpretation $(D, D, \neq, (\iota I))$ where (ιI) equals to I on all object symbols and on all role symbols, but on concept symbols it is 'induced' by ι: $(\iota I)(p) = (I(p), \overline{I(p)})$ for every concept symbol p. The following proposition demonstrates that semantics of $\mathcal{ALC}$ in terminological interpretation (D, I) and in conceptual interpretation $CL(D, D, \neq, (\iota I))$ are closely connected.

Proposition 3. *For every $\mathcal{ALC}$ concept term Z and every terminological interpretation (D, I), the following equality holds: $(\iota I)(Z) = \iota(I(Z))$.*

Proof. Since $\iota(I(Z)) = (I(Z), \overline{I(Z)})$, hence we have to prove that $(\iota I)(Z) = (I(Z), \overline{I(Z)})$. The proof proceeds by induction on structure of a concept term.

Basis of induction: Z is $\bot$, $\top$, or a concept symbol. For concept symbols proof follows from the definition of (ιI). For $\bot$ and $\top$ proofs are similar to each other, so let us present just one of them: $(\iota I)(\top) = sup_{K^{\neq}_D}\{(S, \overline{S}) : S \subseteq D\} = (D, \emptyset)$.

Induction hypothesis: let us assume that $(\iota I)(Z) = (I(Z), \overline{I(Z)})$ for every positive concept term that has at most $k \geq 0$ concept constructs $\neg$, $\sqcup$, $\sqcap$, $\forall$, $\exists$.

Induction step: Z is (1) $(\neg X)$, (2) $(X \sqcup Y)$, (3) $(X \sqcap Y)$, (4) $(\forall R.\ X)$, or (5) $(\exists R.\ X)$, where concept terms X and Y contain at most k constructs $\neg$, $\sqcup$, $\sqcap$, $\forall$, and $\exists$. Let us remark that proofs for cases (2) and (3) are very similar to each other, as well as proofs for cases (4) and (5). So we present proofs for cases (1), (2) and (5) only.

Case $Z \equiv (\neg X)$: Let $(\iota I)(X) = (Ex, In)$. Then $(\iota I)(Z) = (In, Ex)$. By induction hypothesis, $In = \overline{I(X)}$ and $Ex = I(X)$. But $I(Z) = \overline{I(X)}$ and $\overline{I(Z)} = \overline{\overline{I(X)}} = I(X)$. Hence $(\iota I)(Z) = (In, Ex) = (\overline{I(X)}, I(X)) = (I(Z), \overline{I(Z)}$.

Case $Z \equiv (X \sqcup Y)$: $(\iota I)(Z) = sup_{K_D^{\neq}}((\iota I)(X)\ ,\ (\iota I)(Y)) =$ (by induction hypothesis) $= sup_{K_D^{\neq}}((I(X), \overline{I(X)}\ )\ ,\ (I(Y), \overline{I(Y)}\ )) = ((I(X) \cup I(Y))\ ,\ (\overline{I(X)} \cap \overline{I(Y)}\ )) = ((I(X) \cup I(Y))\ ,\ (\overline{I(X) \cup I(Y)}\ )) = (I(Z), \overline{I(Z)})$.

Case $Z \equiv (\exists R.\ X)$: Let $(\iota I)(X) = (Ex', In')$. Then $(\iota I)(Z) =$
$= \sup_{K_D^{\neq}}\{(Ex, In) \in CL(K_D^{\neq}) : \forall o \in Ex\ \forall a \in In\ \exists o' \in D\ \forall a' \in D$
$((a, a') \in I(R) \Rightarrow (o, o') \in I(R),\ o' \in Ex'$, and $a' \in In')\} =$
(by induction hypothesis and
the complement property of the powerset concept lattice)
$= \sup_{K_D^{\neq}}\{(S, \overline{S}) : S \subseteq D$ and $\forall o \in S\ \forall a \in \overline{S}\ \exists o' \in D\ \forall a' \in D$
$((a, a') \in I(R) \Rightarrow (o, o') \in I(R),\ o' \in I(X)$, and $a' \in \overline{I(X)}\ )\}$.
Let us denote the righthandside set of formal concepts by $\mathcal{SX}$. Since $\overline{I(\exists R.X)} = I(\forall R.(\neg X))$, then $(I(\exists R.X), \overline{I(\exists R.X)}) = (I(\exists R.X), I(\forall R.(\neg X))) \in \mathcal{SX}$. Hence $(\iota I)(Z) \succeq_{K_D^{\neq}} (I(\exists R.X),\ I(\forall R.(\neg X)))$. At the same time it follows from proposition 1, that $I(\exists R.X)$ is the greatest subset S of D such that $\forall o \in S\ \exists o' \in D\ ((o, o') \in I(R)\ \&\ o' \in I(X))$. Hence $(\iota I)(Z) = (I(\exists R.X), \overline{I(\exists R.X)}) = (I(Z), \overline{I(Z)})$. ■

Informally speaking, the above proposition states that semantics of $\mathcal{ALC}$ on concept lattices that is defined in Definition 15 is compatible with the standard Kripke-like set-theoretic semantics of $\mathcal{ALC}$ that is given in Definition 3. Due to this interpretation of the proposition, we would like to refer it as the compatibility property, and consider as a strong evidence for a naturalness of Definition 15.

Definition 16. *A concept term X is said to be a tautology, if $I(X) = D$ for every terminological interpretation (D, I). A concept term X is said to be a conceptual tautology, if $I(X) = \top_{(O,A,B)}$ for every conceptual interpretation (O, A, B, I). Conceptual tautology problem for a particular description logic is an algorithmic problem to decide for an input concept term (in the given logic) whether it is a conceptual tautology.*

It follows from the compatibility property that every conceptual tautology is a tautology. But we do not know yet whether the inverse implication holds. We also do not know whether the conceptual tautology problem for $\mathcal{ALC}$ or the positive fragment of $\mathcal{ALC}$ is decidable.

6 Conclusion

The present paper presents a variant of a description logic on concept lattices. It expands a research on an emulation-based combination of Description Logic and Formal Concept Analysis that has been started by a workshop paper [11]. The cited paper has discussed how to emulate basic constructs of Formal Concept Analysis in terms of Description Logic. It has been demonstrated in [11] that FCA can be 'integrated' by Description Logic at least from a viewpoint of 'abstract' expressive power. More formally it can be explained as follows.

Assume that $\mathcal{S}$ is a finite collection of set-theoretic (in)equalities written in terms of uninterpreted symbols for individual objects and attributes, for sets of objects and attributes, for formal contexts and concepts, with aid of set-theoretic operations, upper and lower derivative, intent and extent operations. Then $\mathcal{S}$ can be translated to a description logic knowledge base $KBase(\mathcal{S})$ so that $KBase(\mathcal{S})$ is satisfiable iff there $\mathcal{S}$ is 'consistent', i.e. there exists a formal context that realizes all (in)equalities of $\mathcal{S}$ simultaneously. Since the satisfiability problem is decidable for many description logics, the realization problem for collections of (in)equalities of this kind can be solved (as a rule) automatically (i.e. by some algorithm).

The present paper defines a particular variant of description logic on concept lattices that emulates major Description Logic constructs in terms of operations on concept lattices. The paper demonstrates that one of the basic description logics $\mathcal{ALC}$ with the standard set-theoretic semantics can be naturally interpreted on powerset concept lattices. Hence the proposed approach to Description Logic on concept lattices can be considered compatible with the standard Kripke semantics.

Application of $\mathcal{ALC}$/FCA and FC-$\mathcal{ALC}$ to the knowledge representation and processing will be illustrated with examples and discussed in the full version of publication. Algorithmic and reasoning issues[9] for description logics extended by upper and lower derivatives, for $\mathcal{ALC}$ on concept lattices, interpretation of $\mathcal{ALC}$ on concept lattices in terms of standard description logics are topics for further research.

Acknowledgement. We would like to thank Prof. Karl Erich Wolff, Prof. Bernhard Ganter, and Prof. Franz Baader for discussions at early stage of this research. We are especially obliged to anonymous review of our paper who found out a mistake in Proposition 2, and suggested a correction. We follow this correction in the present version.

References

1. Baader F., Ganter B., Sattler U., and Sertkaya B. *Completing Description Logic Knowledge Bases using Formal Concept Analysis.* In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07). AAAI Press, 2007.
2. Baader F., D. Calvanese, D. Nardi D.McGuinness, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory,Implementation and Applications.* Cambridge University Press, 2003.
3. Bull R.A., Segerberg K. *Basic Modal Logic.* Handbook of Philosophical Logic, v.II, Reidel Publishing Company, 1984 (1-st ed.), Kluwer Academic Publishers, 1994 (2-nd ed.), p.1-88.
4. Ganter B., Wille R. *Formal Concept Analysis.* Mathematical Foundations. Springer Verlag, 1996.
5. Hustadt U, Schmidt R.A., and Georgieva L. *A Survey of decidable first-order fragments and description logics.* J. Relational Methods in Computer Science, Vol. 1, 2004, pp. 251-276.

[9] like decidability, complexity, axiomatization, and model checking

6. Kozen D., Tiuryn J. *Logics of Programs.* Handbook of Theoretical Computer Science, v.B, Elsevier and The MIT Press, 1990, p.789-840.
7. *OWL Web Ontology Language Guide: W3C Recommendation 10 February 2004.* W3C (2004-02-10).
8. Rouane A.H., Huchard M., Napoli A., Valtchev P. *A proposal for combining formal concept analysis and description logics for mining relational data.* Lecture Notes in Artificial Intelligence, Vol. 4390, 2007, pp. 51-65.
9. Rudolph S. *Relational Exploration - Combining Description Logics and Formal Concept Analysis for Knowledge Specification.* PhD Dissertation, Universitatsverlag Karlsruhe, 2006.
10. Schmidt-Schauß M., and Smolka G. *Attributive concept descriptions with complements.* J. Artificial Intelligence, Vol. 48, 1991, pp. 1-26.
11. Shilov N.V. Garanina N.O., Anureev I.S. *Combining Two Formalism for Reasoning about Concepts.* Manuscript. Has been accepted for presentation at 20th International Workshop on Description Logics (DL-2007), Brixen-Bressanone, Italy 8-10 June 2007.
12. Wolff K.E. *A first course in Formal Concept Analysis.* F. Faulbaum StatSoft'93, Gustav Fischer Verlag, 1994, p.429438.
13. Zolin E. Complexity of reasoning in Description Logics. Web-resource available at URL `http://www.cs.man.ac.uk/∼ezolin/logic/complexity.html`.

Horn Representation of a Concept Lattice

Kamel Ben-Khalifa, Susanne Motameny

ZAIK, Center for Applied Computer Science,
University of Cologne, Germany

Abstract. Concept lattices are the central notion of formal concept analysis. They are applied in many different areas such as data mining, knowledge representation or ontology engineering and are subject to ongoing research. In order to understand better the nature of concept lattices it is useful to consider their links to other mathematical notions. For example, a concept lattice can be viewed as a special kind of poset or closure system. In this paper we consider another view of concept lattices by establishing a link to propositional formulas and a special closure property of relations. The main result is an elementary derivation of a Horn formula that uniquely represents a concept lattice based on prime implicates. Using the derived Horn representation, we reestablish the #P-completeness of the concept counting problem and find that the Horn representation is closely related to the stem base of a concept lattice.

1 Introduction

Concept lattices structure the formal concepts of their corresponding formal contexts in a hierarchical manner. They form a central element of formal concept analysis (FCA) and are used in many applications to structure object-attribute data in an informative way. To fully determine a concept lattice, it suffices to know one of the closure systems of concept intents or concept extents (as a formal concept is determined uniquely by its intent or extent, respectively). Our goal in this paper is to derive a conjunctive normal form (CNF) Horn formula which is satisfied exactly by the concept intents of a given context.
Relationships between FCA and Horn formulas have been pointed out by several other authors during the recent years. In [3] Horn functions of general closure systems are constructed (for example the closure systems of concept intents) in order to achieve a translation of certain notions of FCA (in case of [3] that of domination) into a logical framework. For our representation of concept lattices by Horn formulas we decided on different initial conditions and definitions so that the Horn formulas produced by our approach differ from those of [3] in decisive aspects. In [2] the authors show that if a formal context is viewed as a theory in the sense of propositional logic, the attribute implications holding in the context are equivalent to the empirical Horn approximation or Horn least upper bound (as defined in [18]) of that theory. Equivalence here means that the attribute sets respecting all the attribute implications are in one-to-one correspondence

with the tuples of the resulting Horn theory. Thus a link between FCA and knowledge compilation is established. The authors stop at this point, but their result eventually entails an equivalence of concept lattices and certain Horn formulas. In contrast to [2] where the authors introduce an artificial attribute that is satisfied by no object, we can do without such extensions and provide a direct translation of FCA notions into propositional logic.
The paper is organized as follows: In Section 2 we recall the necessary basic definitions and facts from FCA and propositional logic. The derivation of the Horn formula is accomplished in Section 3 and we discuss the relationship of our result to attribute implications and the counting of concepts in Section 4. We close the paper with a summary in Section 5.

2 Basic Definitions and Facts

We now give the basic definitions and facts that we use from formal concept analysis and propositional logic. The notions from propositional logic are well known and only repeated to make the paper self contained. Concerning the FCA terminology we follow the notations from [5].

2.1 Formal Concept Analysis

Let G be a set of objects, M a set of attributes and $I \subseteq G \times M$ an incidence relation between G and M. For $g \in G$ and $m \in M$, gIm is read as "object g has the attribute m" or dually "attribute m is satisfied by object g". The triple $K = (G, M, I)$ is called a *(formal) context.* We consider the usual derivation operators: For $A \subseteq G$ and $B \subseteq M$

$$A' = \{m \in M \mid gIm \ \forall g \in A\}$$
$$B' = \{g \in G \mid gIm \ \forall m \in B\}.$$

A set $B \subseteq M$ is an *intent* of (G, M, I) if and only if $B = B''$. M is always an intent of (G, M, I) as all objects in the empty set trivially have all attributes. Also the set $\{g\}'$ is an intent for each object g and is called *object intent* of $g \in G$. It is well known that every intent of (G, M, I) (except for M) is an intersection of object intents. This fact forms the basis for our derivation of the Horn representation of a concept lattice.

2.2 Propositional Logic

We first introduce the syntactical objects of propositional logic as far as needed for the later discussion and then define their semantics. A lower case letter or an indexed lower case letter, x or x_i, respectively, is a *variable.* We will only use variables that can take values in {0,1} and call these *propositional* *variables*. Furthermore, we use the symbols $\vee, \wedge$, and $^-$ as well as (and). If x is a propositional variable, then $\bar{x}$ is called its *negation*. A *literal* is a propositional variable (called *positive* literal) or its negation (called *negative* literal). A *clause* is a string of the form $(x_1 \vee x_2 \vee \cdots \vee x_n)$, where each $x_i, i \in \{1, \ldots, n\}$ is a literal. A clause is *Horn* if it contains at most one positive literal. It is called *definite Horn* if it contains exactly one positive literal. We will denote the set of literals of a clause C by $\mathrm{lit}(C)$. A *formula in conjunctive normal form (CNF)* is a string of the form

$$C_1 \wedge C_2 \wedge \cdots \wedge C_n,$$

where each $C_i, i \in \{1, \ldots, n\}$ is a clause. If every clause in a CNF formula F is Horn, F is called a *Horn formula*.
Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n propositional variables with a fixed numbering and $L_X = \{x, \bar{x} \ : \ x \in X\}$ be the set of literals over X. An *assignment* is a mapping $\mathcal{A} : X \longrightarrow \{0, 1\}$. An assignment $\mathcal{A}$ can be extended to an assignment $\bar{\mathcal{A}} : L_X \longrightarrow \{0, 1\}$ by setting

$$\bar{\mathcal{A}}(x) = \begin{cases} \mathcal{A}(x) : x \text{ is a positive literal} \\ 1 - \mathcal{A}(x) : x \text{ is a negative literal.} \end{cases} \tag{1}$$

By Equation (1), an assignment $\bar{\mathcal{A}}$ on the set of literals is determined uniquely by an assignment $\mathcal{A}$ on the set of variables. Therefore we will represent an assignment $\bar{\mathcal{A}} : L_X \longrightarrow \{0, 1\}$ by the underlying assignment $\mathcal{A} : X \longrightarrow \{0, 1\}$. With this convention we will use the notation $\mathcal{A}(x)$ for $x \in L_X$ (instead of $\bar{\mathcal{A}}(x)$) throughout the remainder of this paper. An assignment $\mathcal{A}$ *satisfies* a clause C over X if there is at least one literal x in C with $\mathcal{A}(x) = 1$. We then call $\mathcal{A}$ a *model* of C and write $\mathcal{A} \models C$. An assignment $\mathcal{A}$ is called a model of a CNF formula F over X if $\mathcal{A}$ satisfies every clause in F. In this case we write $\mathcal{A} \models F$. The set of all models of a formula F is denoted by $\mathrm{Mod}(F)$. We write $\mathcal{A} \not\models F$ if $\mathcal{A}$ is not a model of F. An assignment $\mathcal{A}$ can be represented by a tuple $t \in \{0, 1\}^n$ by setting $t_i = \mathcal{A}(x_i)$ for each variable x_i. This representation is unique and we will identify assignments and their representing tuples in the following. In particular, we will write $t(x)$ instead of $\mathcal{A}(x)$ and $t \models F$ instead of $\mathcal{A} \models F$. Throughout the remainder of the paper we will refer to a set of tuples $R \subseteq \{0, 1\}^n$ as an *n-ary relation* (or simply *relation* if the arity is not of interest). When we refer to the conjunction of tuples we mean the entry-wise application of the logical $\wedge$-operation, for example

$$(1, 0, 0, 1) \wedge (0, 1, 0, 1) = (0, 0, 0, 1).$$

The essential notion needed in the following is that of a *prime implicate*. We will use prime implicates to construct a CNF formula that is equivalent to a given relation.

Definition prime implicate: Let R be a relation. A clause C is called *implicate* of R if $R \subseteq \mathrm{Mod}(C)$. C is called *prime implicate* if no clause D with $\mathrm{lit}(D) \subset \mathrm{lit}(C)$ is an implicate of R.

It is a well known fact that the CNF formula F that consists of the conjunction of all prime implicates of a relation R is equivalent to R in the following sense:

$$t \in R \Longleftrightarrow t \models F \ (\text{that is } R = \mathrm{Mod}(F)\,) \tag{2}$$

3 Derivation of the Horn Representation

With the definitions from Section 2 we are now ready to derive the Horn representation of a concept lattice. We first note a special property of Horn clauses.

Lemma 1: Let $t^{(i)}, i \in \{1, \ldots, k\}$ be k n-ary tuples and C a Horn clause in the propositional variables $x_1, x_2, \ldots, x_n$. It holds that

$$\forall i \in \{1, \ldots, k\}\ t^{(i)} \in \mathrm{Mod}(C) \implies \bigwedge_{i=1}^{k} t^{(i)} \in \mathrm{Mod}(C).$$

This is a well known fact which follows for example from [1], Lemma 3. A relation R is called $\wedge$-*closed* if with every two tuples $t^{(1)}, t^{(2)} \in R$ also $t = t^{(1)} \wedge t^{(2)}$ is an element of R. We continue with an important and well known fact about $\wedge$-closed relations which dates back to [10, 15] and can also be found in [9], [11], and [13] in a more general form than the one we prove. Our formulation however is sufficient and more appropriate for our considerations.

Proposition 1: R is $\wedge$-closed if and only if all prime implicates of R are Horn clauses.

Proof: Let R be $\wedge$-closed and C an arbitrary prime implicate of R. Assume that C contains at least two positive literals y and z. If we denote the disjunction of the remaining literals by D we have $C = (y \vee z \vee D)$. By definition $R \subseteq \mathrm{Mod}(C)$. If every tuple in R was a model of D, then D would be an implicate of R. This cannot happen because C is a prime implicate. Therefore R contains a nonempty set of tuples T with $T \cap \mathrm{Mod}(D) = \emptyset$. To fulfill the condition $t \models C$ for a $t \in T$, either $t(y) = 1$ or $t(z) = 1$ must hold. If we had $t(y) = 1$ (or analogously $t(z) = 1$) for all $t \in T$, then $(y \vee D)$ (or $(z \vee D)$, respectively) would be an implicate of R and C could not be a prime implicate. This implies the existence of tuples $t^{(y)}$ with $t^{(y)}(y) = 1$ and $t^{(y)}(z) = 0$ and $t^{(z)}$ with $t^{(z)}(y) = 0$ and $t^{(z)}(z) = 1$ in T. Let $t^* = t^{(y)} \wedge t^{(z)}$. For t^* it holds that

$$\begin{aligned} t^* &\not\models D \\ t^*(y) &= 0 \\ t^*(z) &= 0 \end{aligned}$$

and therefore $t^* \notin \mathrm{Mod}(C)$. This means that $t^* \notin R$, which is a contradiction to the precondition that R is $\wedge$-closed.
Now let R be a relation whose prime implicates $C_j, j \in \{1, \ldots, k\}$ are all Horn. Consider two arbitrary tuples $t^{(1)}$ and $t^{(2)}$ in R. Let $t^* = t^{(1)} \wedge t^{(2)}$. According to Lemma 1 $t^{(1)} \in \mathrm{Mod}(C_j)$ and $t^{(2)} \in \mathrm{Mod}(C_j)$ implies $t^* \in \mathrm{Mod}(C_j)$ for all $j \in \{1, \ldots, k\}$. From $R = \mathrm{Mod}(\bigwedge_{j=1}^{k} C_j) = \bigcap_{j=1}^{k} \mathrm{Mod}(C_j)$ it follows that $t^* \in R$ which proves the $\wedge$-closedness of R. □

For the purpose of our derivation of a Horn representation of a concept lattice we now provide an obvious translation of a formal context $K = (G, M, I)$ into a $|M|$-ary relation. Let $m_1, m_2, \ldots, m_{|M|}$ be the attributes in the cross table of K. We associate with every attribute m_i a propositional variable x_i and replace the crosses in the table by ones and the empty cells by zeroes. The row corresponding to an object $g \in G$ now represents a tuple from $\{0, 1\}^{|M|}$ denoted by t^g. Finally, the tuple t^M which contains only ones is added to the table. The relation gained from a context K by this translation is denoted by R_K. Table 1 illustrates the translation by an example.

K	m_1	m_2	m_3	m_4
a		X	X	
b				X
c		X		X
d	X	X		X

R_K	x_1	x_2	x_3	x_4
t^a	0	1	1	0
t^b	0	0	0	1
t^c	0	1	0	1
t^d	1	1	0	1
t^M	1	1	1	1

Table 1. A formal context K and its corresponding relation R_K

Remark: By adding t^M to R_K, we ensure that we arrive at a definite Horn representation (called pure Horn representation in [3]), that is each clause in the representation contains exactly one positive literal. This is what makes our approach different from [3] and turns out to yield a very nice correspondence to attribute implications as described in Section 5.

The result of our translation procedure is a $|M|$-ary relation R_K which contains one tuple for each object intent as well as the tuple t^M that corresponds to the intent consisting of all attributes. Recall that all other concept intents are intersections of object intents. The intersection of object intents translates exactly to the conjunction of their corresponding tuples. Consider for example the objects a and c from Table 1. Their object intents are $\{a\}' = \{m_2, m_3\}$ and $\{c\}' = \{m_2, m_4\}$. The concept intent resulting from their intersection corresponds to the tuple $t = t^a \wedge t^c$:

$$\begin{array}{rcccc} \{a\}' \cap \{c\}' & = & \{m_2, m_3\} \cap \{m_2, m_4\} & = & \{m_2\} \\ t^a \wedge t^c & = & (0,1,1,0) \wedge (0,1,0,1) & = & (0,1,0,0) \end{array}$$

With this procedure we can inductively construct to each concept intent a corresponding tuple by simply computing the conjunction of the tuples belonging to the involved object intents. We add all thus constructed tuples to R_K and call the resulting relation $\hat{R}_K$. Obviously there is a bijection between concept intents of $K = (G, M, I)$ and tuples in $\hat{R}_K$. Also, because we collected all possible conjunctions of tuples from R_K in $\hat{R}_K$, $\hat{R}_K$ is $\wedge$-closed. In fact it is exactly the $\wedge$-closure of R_K. By Proposition 1, $\hat{R}_K$ can be represented by the Horn formula F_K consisting of its prime implicates. This formula is the Horn representation of the concept lattice we wanted to derive.

Definition Horn representation: Let $C_1, C_2, \ldots, C_k$ be the prime implicates of $\hat{R}_K$. The formula

$$F_K = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$

is called *Horn representation* of the concept lattice defined by $K = (G, M, I)$.

Note that we use all prime implicates of $\hat{R}_K$ in the above definition. In general, F_K can contain redundant prime implicates. If these are removed, the obtained formula is still equivalent to $\hat{R}_K$ in the sense of (2). We use all prime implicates in our definition of a Horn representation for reasons of uniqueness. The Horn representation of the context from Table 1 for example is

$$F_K = (\bar{x}_1 \vee x_2) \ \wedge \ (\bar{x}_1 \vee x_4) \ \wedge \ (\bar{x}_3 \vee x_2) \ \wedge \ (\bar{x}_3 \vee \bar{x}_4 \vee x_1).$$

The tuples of $\hat{R}_K$ correspond to the concept intents of K bijectively. The hierarchical order of the concept intents is given by $B_1 \leq B_2 \iff B_1 \supseteq B_2$ for two concept intents B_1 and B_2 (see [5]). It corresponds to the following order on $\hat{R}_K$:

$$t^{(1)} \leq t^{(2)} \iff t^{(1)} \wedge t^{(2)} = t^{(2)}$$

The concept extent corresponding to the intent represented by the tuple $t \in \hat{R}_K$ is denoted by t' and can be determined by

$$t' = \{g \in G \mid t^g \wedge t = t\}.$$

In theory, F_K can be computed from R_K directly, that is one does not need to determine the (possibly exponentially sized) $\wedge$-closure of R_K.

Proposition 2: The prime implicates of $\hat{R}_K$ are exactly the Horn prime implicates of R_K.

Proof: Let C be a Horn prime implicate of R_K. Because of $R \subseteq \mathrm{Mod}(C)$ it follows from Lemma 1 that $\hat{R}_K \subseteq \mathrm{Mod}(C)$. Therefore C is an implicate of $\hat{R}_K$. C is also a prime implicate of $\hat{R}_K$, because for all D with $\mathrm{lit}(D) \subset \mathrm{lit}(C)$ there is a tuple $t \in R_K$ (and in particular $t \in \hat{R}_K$) satisfying $t \not\models D$.

Let now conversely C be a prime implicate of $\hat{R}_K$. According to Proposition 1 C is Horn. Because of $\hat{R}_K \supseteq R_K$, C is an implicate of R_K. Assume that C is no prime implicate of R_K. Then there is an implicate D of R_K with $\text{lit}(D) \subset \text{lit}(C)$. C is a prime implicate of $\hat{R}_K$ and therefore there is a tuple $t \in \hat{R}_K$ that is no model for D. This tuple is a conjunction of tuples $t^{(i)} \in R_K$ and as C is Horn so is D. Lemma 1 implies the existence of a tuple $t^{(i^*)} \in R_K$ with $t^{(i^*)} \notin \text{Mod}(D)$. But this is a contradiction to our presupposition of D being an implicate of R_K, which completes the proof. $\square$

Proposition 2 shows that it suffices to compute the prime implicates of R_K, select those that are Horn, and their conjunction is exactly the Horn representation F_K. However, there are several complexity results from propositional logic that imply limitations for the practical implementation of such a procedure. From a result shown in [12] it follows that there exist contexts $K = (G, M, I)$ containing only join-irreducible objects (called characteristic models in [12]) so that the shortest Horn formula equivalent to $\hat{R}_K$ is exponential in $|G|$. Thus a method running in input polynomial time that computes the Horn representation of any concept lattice from its context does not exist. Another result in [8] states that a Horn formula that is equivalent to an $\wedge$-closed n-ary relation R can be found in $O(|R|\,n\;(|R| + n))$ time. Moreover, the constructed formula contains no more than $|R|\,n$ clauses. This means that we can obtain a Horn formula that is equivalent to F_K in $O(|\hat{R}_K|\,|M|\;(|\hat{R}_K| + |M|))$ time. However, the input to this algorithm would be the relation $\hat{R}_K$ which may be exponentially larger than R_K. Thus, the algorithm from [8] cannot be used to generate a Horn formula representing the concept lattice of a given context in time polynomial in the size of the context. As in general large relations can be equivalent to short Horn formulas, the time complexity of such an approach would not even be polynomial relative to the input and output size. In [13] it is proved that the computation of a Horn formula that is equivalent to $\hat{R}_K$ and takes R_K as input is at least as difficult as the generation of all the transversals of a hypergraph. To our knowledge the complexity of this hypergraph problem is still open.

4 Discussion

In this section we discuss the relationship of our result to certain areas and problems of Formal Concept Analysis.

4.1 Counting of Concepts

In the previous section we derived a Horn formula F_K that represents the intent set of a formal context $K = (G, M, I)$ and therefore is also a representation of its concept lattice. If we count the models of F_K we actually count the concepts of $K = (G, M, I)$ and our result thus offers a possibility to determine the size of a concept lattice. However, in [4] a theorem is proven which implies that the counting of the models of a definite Horn formula is $\#P$-complete in general, that is for a definite Horn formula there is a nondeterministic Turing machine whose number of accepting computation paths equals the number of models of that formula and that runs in polynomial time (see for example [16] for a rigorous definition of the complexity class $\#P$). This is in accordance with [14], where the $\#P$-completeness of the determination of the size of a concept lattice is also proved. The fact that the counting of concepts is $\#P$-complete also follows easily from the $\#P$-completeness of the counting of ideals of a poset which was proved in [17]. If $(P, \leq)$ is an arbitrary poset, then the formal context $(P, P, \not\leq)$ contains as many concepts as there are ideals in $(P, \leq)$. This also answers the question raised in [14] whether the $\#P$-completeness also holds for distributive concept lattices because the concept lattice of $(P, P, \not\leq)$ is always distributive.

4.2 Attribute Implications

We defined R_K in such a way that $t^M \in R_K$. This causes all clauses of F_K to contain exactly one positive literal and they can be equivalently written as implications:

$$(\bar{x}_1 \vee \bar{x}_2 \vee \cdots \vee \bar{x}_k \vee y) \Longleftrightarrow x_1 \wedge x_2 \wedge \cdots \wedge x_k \longrightarrow y$$

If we write instead of the propositional variable x_i the attribute m_i which it represents, we can translate our Horn representation to a set of attribute implications. We recall some basic definitions from FCA concerning attribute implications (see [5]).

Definition attribute implications: The implication $X \longrightarrow Y$ is equivalent to $Y \subseteq X''$. A set $T \subseteq M$ is said to *respect* an implication $X \longrightarrow Y$ if $X \nsubseteq T$ or $Y \subseteq T$. A set of attributes T respects a set of implications $\mathcal{L}$ if it respects every single implication in $\mathcal{L}$.
An implication $X \longrightarrow Y$ *follows* from an implication set $\mathcal{L}$ if every set $T \subseteq M$ that respects $\mathcal{L}$ also respects $X \longrightarrow Y$. An implication $X \longrightarrow Y$ *holds* in a context (G, M, I) if every concept intent of (G, M, I) respects the implication $X \longrightarrow Y$.

Because all concept intents of $K = (G, M, I)$ respect all the implications obtained from the prime implicates of $\hat{R}_K$ by definition, the set of all these implications is complete for K, that is all attribute implications holding in K follow from it. Merging implications with the same premise which can generally occur in F_K, we obtain the set $\mathcal{H}$ of attribute implications. We illustrate this procedure using the Horn representation of the context K from Table 1:

Horn Representation	Implications	Merged Implications
$F_K = \bar{x}_1 \vee x_2 \wedge$	$\{m_1 \longrightarrow m_2$	$\mathcal{H} = \{m_1 \longrightarrow m_2, m_4$
$\bar{x}_1 \vee x_4 \wedge$	$m_1 \longrightarrow m_4$	$m_3 \longrightarrow m_2$
$\bar{x}_3 \vee x_2 \wedge$	$m_3 \longrightarrow m_2$	$m_3, m_4 \longrightarrow m_1\}$
$\bar{x}_3 \vee \bar{x}_4 \vee x_1$	$m_3, m_4 \longrightarrow m_1\}$	

Unfortunately $\mathcal{H}$ can contain redundant implications because the set of prime implicates it is based on can be redundant. Recall that an implication base of K is a set of implications that is complete for K and minimal in the sense that it does not contain redundant implications. Therefore, if the redundant implications are removed from $\mathcal{H}$ an implication base of K is the result. In FCA the usual implication base of interest is the stem base or Duquenne-Guigues base first described in [6]. Its definition is based on the notion of a pseudo intent (see [5]).

Definition pseudo intent: A subset $P \subseteq M$ of attributes is called *pseudo intent* of a context (G, M, I) if

(1) $P \neq P''$
(2) $\forall Q \subset P, Q$ pseudo intent $\Longrightarrow Q'' \subset P$.

In the above definition M is considered to be finite. The stem base is then defined as:

Definition stem base: The set of implications

$$\mathcal{L} = \{P \longrightarrow P'' \mid P \text{ is a pseudo intent of } (G, M, I)\}$$

is an implication base of the context (G, M, I) and is called *stem base* of the context.

When comparing $\mathcal{H}$ to the stem base $\mathcal{L}$ of $K = (G, M, I)$, we can see immediately that for each premise P of the stem base there must be a premise X of $\mathcal{H}$ satisfying $X \subseteq P$. This is because the premises in $\mathcal{H}$ correspond to prime implicates and every implication holding in $K = (G, M, I)$ is an implicate of $\hat{R}_K$ and thus must contain a prime implicate. We can see this already in our simple example:

$$\begin{array}{l|l} \mathcal{H} = \{m_1 \longrightarrow m_2, m_4 & \mathcal{L} = \{m_1 \longrightarrow m_2, m_4 \\ \quad m_3 \longrightarrow m_2 & \quad m_3 \longrightarrow m_2 \\ \quad m_3, m_4 \longrightarrow m_1\} & \quad m_2, m_3, m_4 \longrightarrow m_1\} \end{array}$$

The premise of the last implication in $\mathcal{L}$ is a superset of the premise of the last implication in $\mathcal{H}$. We have already seen that each pseudo intent contains the premise of an implication in $\mathcal{H}$. Moreover, we have the following conjecture.

Conjecture: Each premise in $\mathcal{H}$ is contained in a pseudo intent.

We believe that this conjecture is true but could not find a rigorous proof for this claim up to now. However, Proposition 3 shows that there is a subset of premises in $\mathcal{H}$ each of which is not only contained in a pseudo intent but is a pseudo intent itself.

Proposition 3: Let $X \longrightarrow Y$ be an implication in $\mathcal{H}$. If $X \cup Y$ is closed (that is $(X \cup Y)'' = X \cup Y$) then X is a pseudo intent.

Proof : Let $R \subset X$. Because $X \longrightarrow y$ is a prime implicate for all $y \in Y$ it follows that $R \not\longrightarrow y$ and thus $R'' \cap Y = \emptyset$. Because $X \longrightarrow Y$ is equivalent to $Y \subseteq X''$ we obtain $X \cup Y \subseteq X''$. On the other hand we have $X'' \subseteq (X \cup Y)'' = X \cup Y$ and thus $X'' = X \cup Y$ which implies $R'' \subset X$. Therefore X satisfies the definition of a pseudo intent. $\square$

For the case when $X \cup Y$ is not closed we have not been able to prove that X is contained in a pseudo intent nor could we construct a counterexample. Thus, the question whether our conjecture is true or not is currently open.
We close this section with a remark about the minimality of the Duquenne-Guigues base. The comparison of $\mathcal{H}$ and $\mathcal{L}$ from our example context shows that both contain the same number of implications but the last implication in $\mathcal{H}$ has a shorter premise. In general, the stem base $\mathcal{L}$ allows for a more compact representation (that is one that uses less attributes), which can be computed from $\mathcal{L}$ in the following way: First $\mathcal{L}$ is translated into a Horn formula $F_{\mathcal{L}}$ (for example by using the translation of F_K to $\mathcal{H}$ demonstrated above backwards). Then for each clause it is checked whether it contains a subclause that is an implicate of $F_{\mathcal{L}}$. If so, the clause is replaced by that subclause. When iterated, this procedure yields a formula consisting of prime implicates that is equivalent to $\mathcal{L}$ and the translation back into a set of implications is the desired compact representation of $\mathcal{L}$. The basics of such a method that requires at most quadratic time in the length of the input formula $F_{\mathcal{L}}$ can be found for example in [7].

5 Summary

We suggested a representation of a concept lattice by a special Horn formula. This formula was constructed as a conjunction of the prime implicates of a relation R_K that was derived from the formal context $K = (G, M, I)$. Theoretically, the Horn representation can be computed from the context directly but several results from propositional logic imply limitations for the practical implementation. We believe that despite of this drawback our result is interesting insofar as it reveals a close relationship between FCA and propositional logic that can be exploited to tackle research questions in both scientific areas. By using the Horn representation we reproduced the $\#P$-completeness of the concept counting problem. Also we pointed to a close relationship between the Horn representation and the stem base of a formal context.
In our future research we hope to find more links between FCA and propositional logic so that one can use the knowledge and formalism in the one area to solve problems and gain insights in the other one.

6 Acknowledgements

We would like to thank Ulrich Faigle and three anonymous referees whose suggestions helped to improve the paper.

References

1. D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.
2. J. L. Balcázar and J. Baixeries. Discrete deterministic data mining as knowledge compilation. *Workshop Discr Math and Data Mining at SIAM DM Conference*, 2003.
3. A. Berry, E. SanJuan, and A. Sigayret. Generalized domination in closure systems. *Discrete Applied Mathematics*, 154:1064–1084, 2006.
4. N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.
5. B. Ganter and R. Wille. *Formal Concept Analysis, Mathematical Foundations.* Springer, 1999.
6. J.-L. Guigues and V. Duquenne. Familles minimales d'implications informatives resultant d'un tableau de données binaires. *Math. Sci. Humaines*, 95:5–18, 1986.
7. P. L. Hammer and A. Kogan. Horn functions and their DNFs. *Information Processing Letters*, 44:23–29, 1992.
8. J.-J. Herbrard and B. Zanuttini. An efficient algorithm for Horn description. *Information Processing Letters*, 88:177–182, 2003.
9. J. Hooker. *Logic-Based Methods for Optimization.* John Wiley & Sons, 2000.
10. A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.
11. H. A. Kautz, M. J. Kearns, and B. Selman. Reasoning with characteristic models. In *National Conference on Artificial Intelligence*, pages 34–39, 1993.
12. H. A. Kautz, M. J. Kearns, and B. Selman. Horn approximations of empirical data. *Artificial Intelligence*, 74(1):129–145, 1995.
13. D. Kavvadias, C. H. Papadimitriou, and M. Sideri. On Horn envelopes and hypergraph transversals. In K. W. Ng, P. Raghavan, N. V. Balasubramanian, and F. Y.L. Chin, editors, *Algorithms and Computation, Proceedings of the 4th International Symposium on Algorithms and Computation*, volume 762 of *Lecture Notes in Computer Science*, pages 399–405. Springer, 1993.
14. S. O. Kuznetsov. On computing the size of a lattice and related decision problems. *Order*, 18:313–321, 2001.
15. J. C. C. McKinsey. The decision problem for some classes without quantifiers. *Journal of Symbolic Logic*, 8:61–76, 1943.
16. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
17. J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal of Computing*, 12(4):777–788, 1983.
18. B. Selman and H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

Formal concepts as optimal factors in Boolean factor analysis: implications and experiments*

Radim Belohlavek[1,2], Vilem Vychodil[2]

[1] Dept. Systems Science and Industrial Engineering
T. J. Watson School of Engineering and Applied Science
Binghamton University–SUNY, PO Box 6000, Binghamton, NY 13902–6000, USA
rbelohla@binghamton.edu

[2] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
vilem.vychodil@upol.cz

Abstract. Boolean factor analysis aims at decomposing an objects $\times$ attributes Boolean matrix I into a Boolean product of an objects $\times$ factors Boolean matrix A and a factors $\times$ attributes Boolean matrix B, with the number of factors as small as possible. This paper is a continuation of our previous paper where we proved that formal concepts of I are optimal factors for Boolean factor analysis. In particular, we concentrate on the implications of the proof. Namely, on the fact that finding factors can be reduced to the set covering problem for which there exist efficient approximation algorithms. In this paper, we present the algorithm for finding factors which results this way and present several experiments on factorizing Boolean matrices.

1 Introduction and problem setting

The present paper concerns factor analysis of Boolean data and is a continuation of [4]. In [4], we proved that formal concepts are optimal factors in Boolean factor analysis and outlined some implications of the insight provided by the proof. The aim of this paper is to elaborate on one of those implications. Namely, on the fact that the problem of finding factors in Boolean factor analysis can be reduced to the well-known set covering problem. For the set covering problem, there exists an efficient approximation algorithm. This algorithm can thus be used for finding factors in Boolean factor analysis. Moreover, the algorithm can be sped up due to some specific features of Boolean factor analysis. In this paper, we present the thus resulting algorithm for finding factors in Boolean factor analysis. The main focus of the paper is on presenting examples on on Boolean factor analysis and experiments with the algorithm.

The idea of factor analysis is rooted in Spearman's monumental development of a psychological theory involving a single general factor and a number of specific factors [18]. Today, factor analysis is a well-established branch of statistical

* Supported by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, and by institutional support, research plan MSM 6198959214.

data analysis with applications in numerous fields and with support in several software packages, see e.g. [1, 7, 10]. According to Harman [10, p. 4], "The principal concern of factor analysis is the resolution of a set of variables linearly in terms of (usually) a small number of categories or 'factors'. ... A satisfactory solution will yield factors which convey all the essential information of the original set of variables. Thus, the chief aim is to attain scientific parsimony or economy of description."

The problem of factor analysis can be described as follows. Suppose we are given an $n \times m$ matrix I describing relationships between n objects and m variables. Each entry I_{ij} of the objects$\times$variables matrix I contains the value of j-th variable on i-th object. The aim is to find k new variables, called factors, an $n \times k$ matrix A describing a relationship between objects and factors, and a $k \times m$ matrix B describing a relationship between factors and original variables, in such a way that I can be obtained from A and B. In case of linear factor analysis, I is required to be (approximately) equal to the usual matrix product $A \circ B$ of A and B. In addition, one requires that the number k of factors be less than the number m of original variables, attaining thus the dimension reduction of the space in which the objects are described. Every A's entry A_{il}, called factor loading, represents a value of l-th factor on i-th object; every B's entry B_{lj}, called factor score, represents the manifestation of j-th variable on l-th factor.

Several extensions of linear factor analysis have been proposed to deal with data for which linear resolution of the original variables in terms of factors is not appropriate, see e.g. [7, 10]. A particular example of such data is represented by Boolean variables (attributes), called also binary variables (attributes), yes-or-no variables (attributes). In this case, entries of I are 1s and 0s, i.e. $I_{ij} = 1$ or $I_{ij} = 0$ with 1/0 indicating that the i-th object has/does not have the j-th variable (attribute). That is, I is a so-called Boolean matrix. For instance, a patient (object) has or does not have headache (variable, attribute). The question of whether the methods of factor analysis are appropriate for Boolean variables has been discussed since 1940s, see e.g. Section 7 of [13]. It has been argued that common methods of factor analysis, both linear and non-linear are not appropriate to handle Boolean variables, see e.g. [13, 16, 19].

A promising way to reveal factors in Boolean data is provided by Boolean factor analysis (BFA), see e.g. [8, 14, 17]. In BFA, a decomposition of an $n \times m$ matrix I with $I_{ij} \in \{0, 1\}$ is sought into a Boolean matrix product $A \circ B$ of an $n \times k$ matrix A with $A_{il} \in \{0, 1\}$ and a $k \times m$ matrix B with $B_{lj} \in \{0, 1\}$ with k as small as possible. Note that a Boolean matrix product $A \circ B$ is defined by

$$(A \circ B)_{ij} = \bigvee_{l=1}^{k} A_{jl} \cdot B_{lj}. \qquad (1)$$

where $\bigvee$ denotes maximum. Using Boolean matrices for the objects$\times$factors and the factors$\times$variables relationships, and using Boolean matrix product has the following advantage:

– One does not have to deal with the problem of rounding off real values to 0 and 1 which is the case when using common methods of factor analysis.

– Clear interpretability of factors. Namely, with the Boolean matrix product, $I = A \circ B$ says: "an object i has an attribute j if and only if there is a factor l such that l applies to i and j is one of the manifestations of l".

Several methods for BFA can be found in the literature. Perhaps the most advanced one is based on using Hopfield-like associative neural networks, see [8, 17]. In this approach, factors correspond to stable points (attractors) of an associative neural network. [12] presents a different approach which is based on using genetic algorithms for the search of factors. Yet another approach, which served as an inspiration for our paper is presented in [11] where the authors try to exploit methods of formal concept analysis [5, 9] for BFA.

In [4], we presented several results on factorizing a Boolean matrix I using formal concepts associated to I. The main result says that formal concepts are optimal factors in BFA, meaning that for every decomposition of I into $A \circ B$ with k factors, i.e., A and B are $n \times k$ and $k \times m$ Boolean matrices, there exists a decomposition of I which uses formal concepts as factors such that the number of the formal concepts is at most k. In addition, we pointed out in [4] that the problem of finding a smallest set of factors in BFA can be reduced to set covering problem for which there exists an efficient approximation algorithm.

In this paper, we present the thus resulting algorithm for finding factors in BFA. The main aim of this paper is to present examples on BFA and experiments with the algorithm.

2 Formal concepts as optimal factors and the reduction to set covering problem

Recall first basic notions and notation from formal concept analysis (FCA) [5, 9]. Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ be sets of objects and attributes, and I be a binary relation between X and Y. The triplet $\langle X, Y, I\rangle$ is called a formal context. Using the well-known fact that a binary relation between X and Y can be represented by an $n \times m$ Boolean matrix, we denote both the binary relation and its corresponding Boolean matrix by I. That is, for the entry I_{ij} of (matrix) I we have $I_{ij} = 1$ iff $\langle x_i, y_j\rangle$ belongs to (relation) I and $I_{ij} = 0$ if $\langle x_i, y_j\rangle$ do not belong to (relation) I. A formal concept of $\langle X, Y, I\rangle$ is any pair $\langle C, D\rangle$ of sets $C \subseteq X$ and $D \subseteq Y$ such that $C = D^{\downarrow}$ and $D = C^{\uparrow}$ where

$$D^{\downarrow} = \{x \in X \mid \text{for each } y \in D : \langle x, y\rangle \in I\}$$

is the set of all objects sharing all attributes from D, and

$$C^{\uparrow} = \{y \in Y \mid \text{for each } x \in C : \langle x, y\rangle \in I\}$$

is the set of all attributes shared by all objects from C. The set of all formal concepts of $\langle X, Y, I\rangle$ is denoted by $\mathcal{B}(X, Y, I)$. Under a natural partial order, $\mathcal{B}(X, Y, I)$ happens to be a complete lattice, so-called concept lattice of $\langle X, Y, I\rangle$ [9, 20]. It is a well-known fact that formal concepts of $\langle X, Y, I\rangle$ are just maximal rectangles of matrix I which are full of 1s. For instance,

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{1} & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

is a Boolean matrix representing a formal context with $X = \{x_1, \ldots, x_4\}$, $Y = \{y_1, \ldots, y_5\}$, and relation I for which $\langle x_1, y_1\rangle \in I$, $\langle x_1, y_2\rangle \in I$, $\langle x_1, y_3\rangle \notin I$, etc. The bold **1**s form a maximal rectangle, with rows 1, 2, 3, and columns 1 and 2. Correspondingly therefore, $\langle\{x_1, x_2, x_3\}, \{y_1, y_2\}\rangle$ is a formal concept in $\langle X, Y, I\rangle$. This "geometrical" way of looking at formal concepts proves to quite useful in FCA.

Let

$$\mathcal{F} = \{\langle A_1, B_1\rangle, \ldots, \langle A_k, B_k\rangle\} \subseteq \mathcal{B}(X, Y, I),$$

i.e. $\mathcal{F}$ is a set of formal concepts associated to I. Denote by $A_\mathcal{F}$ and $B_\mathcal{F}$ the $n \times k$ and $k \times m$ Boolean matrices defined by

$$(A_\mathcal{F})_{il} = \begin{cases} 1 & \text{if } x_i \in A_l, \\ 0 & \text{if } x_i \notin A_l; \end{cases} \quad \text{and} \quad (B_\mathcal{F})_{lj} = \begin{cases} 1 & \text{if } y_j \in B_l, \\ 0 & \text{if } y_j \notin B_l. \end{cases}$$

That is, the l-th column $(A_\mathcal{F})_{\_l}$ of $A_\mathcal{F}$ consists of the characteristic vector of A_l and the l-th row $(B_\mathcal{F})_{l\_}$ of $B_\mathcal{F}$ consists of the characteristic vector of B_l.

We are interested in a decomposition of I into $A_\mathcal{F} \circ B_\mathcal{F}$. If $I = A_\mathcal{F} \circ B_\mathcal{F}$, $\mathcal{F}$ can be seen as a set of factors and we call the formal concepts from $\mathcal{F}$ factor concepts. The l-th factor, i.e. formal concept $\langle A_l, B_l\rangle$, applies to the i-th object if $x_i \in A_l$; the j-th attribute y_j is a manifestation of the l-the factor if $y_j \in B_l$. Note that decomposing I by means of formal concepts has been proposed in [11]. However, the particular way of using formal concepts as factors is not described explicitly in [11].

Two questions then arise. First, does there always exist a $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ such that $I = A_\mathcal{F} \circ B_\mathcal{F}$? Second, to what extent is it optimal to consider formal concepts from $\mathcal{B}(X, Y, I)$ as factors?

For the first question, it is a well-known fact of formal concept analysis that if we put $\mathcal{F} = \mathcal{B}(X, Y, I)$ (with any indexing of formal concepts), then $I = A_\mathcal{F} \circ B_\mathcal{F}$. In this sense, factorization using formal concepts is universal, see [4]. Note that putting $\mathcal{F} = \mathcal{B}(X, Y, I)$ is not practically useful because the number $|\mathcal{B}(X, Y, I)|$ of all formal concepts is usually larger than the number $|Y|$ of the original attributes. A better way consists in taking $\mathcal{F} = \mathcal{O}(X, Y, I)$ or $\mathcal{F} = \mathcal{A}(X, Y, I)$ where

$$\mathcal{O}(X, Y, I) = \{\langle\{x\}^{\uparrow\downarrow}, \{x\}^{\uparrow}\rangle \mid x \in X\} \text{ and } \mathcal{A}(X, Y, I) = \{\langle\{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow}\rangle \mid y \in Y\}$$

are the sets of object-concepts and attribute-concepts, respectively. In both of these cases we have $I = A_\mathcal{F} \circ B_\mathcal{F}$ but still, this may not be the optimal decomposition.

The second question is answered by the following theorem which was proven in [4]. The theorem says that for any decomposition of I there is always at least as good a decomposition which uses formal concept as factors.

Theorem 1 (formal concepts are optimal factors). *Let $I = A \circ B$ for $n \times k$ and $k \times m$ Boolean matrices A and B. Then there exists $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ with*

$$|\mathcal{F}| \leq k$$

such that for the $n \times |\mathcal{F}|$ and $|\mathcal{F}| \times m$ Boolean matrices $A_\mathcal{F}$ and $B_\mathcal{F}$ we have

$$I = A_\mathcal{F} \circ B_\mathcal{F}.$$

The proof is instructive and we therefore summarize its main points. First, observe that $I = A \circ B$ means that I can be written as a $\bigvee$-superposition of rectangles consisting of 1s. For instance, in case of $I = A \circ B$ being

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&1\\ 1&0&1&0\\ 1&1&0&0\\ 0&0&1&0 \end{pmatrix} \circ \begin{pmatrix} 1&1&0&0&0\\ 0&0&1&1&0\\ 1&0&0&0&1\\ 0&1&0&0&0 \end{pmatrix},$$

the corresponding decomposition can be rewritten as a $\bigvee$-superposition

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&0\\ 1&1&0&0&0\\ 0&0&0&0&0 \end{pmatrix} \vee \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&1&0\\ 0&0&0&0&0 \end{pmatrix} \vee \begin{pmatrix} 0&0&0&0&0\\ 1&0&0&0&1\\ 0&0&0&0&0\\ 1&0&0&0&1 \end{pmatrix} \vee \begin{pmatrix} 0&1&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{pmatrix}$$

of rectangles I^1, I^2, I^3, I^4, i.e. Boolean matrices whose 1s form rectangles. The l-th Boolean matrix I^l results as a Boolean matrix multiplication of the l-the column of A ($n \times 1$ matrix) and the l-th row of B ($1 \times m$ matrix). Second, each such a rectangle is contained in a maximal rectangle of I, i.e. in a formal concept, and a $\bigvee$-superposition of these maximal rectangles gives I. Denoting therefore the collection of all these formal concepts by $\mathcal{F}$ yields the conclusion. Note that since two distinct rectangles may be contained in a single maximal rectangle, we may have $|\mathcal{F}| < k$.

Using our example, consider formal concepts $\langle A_1, B_1 \rangle = \langle \{x_1, x_2, x_3\}, \{y_1, y_2\} \rangle$, $\langle A_2, B_2 \rangle = \langle \{x_3\}, \{y_1, y_2, y_3, y_4\} \rangle$, $\langle A_3, B_3 \rangle = \langle \{x_2, x_4\}, \{y_1, y_5\} \rangle$, of I. Then, each of the rectangles corresponding to I^ls is contained in some of the maximal rectangles corresponding to $\langle A_1, B_1 \rangle$, $\langle A_2, B_2 \rangle$, or $\langle A_3, B_3 \rangle$. Putting now $\mathcal{F} = \{\langle A_1, B_1 \rangle, \langle A_2, B_2 \rangle, \langle A_3, B_3 \rangle\}$, we have $I = A_\mathcal{F} \circ B_\mathcal{F}$. Denoting by $(A_\mathcal{F})_{\_l}$ and $(B_\mathcal{F})_{l\_}$ the l-th column of $A_\mathcal{F}$ and the l-th row of $B_\mathcal{F}$, $I = A_\mathcal{F} \circ B_\mathcal{F}$ can further be rewritten as $I = (A_\mathcal{F})_{\_1} \circ (B_\mathcal{F})_{1\_} \vee (A_\mathcal{F})_{\_2} \circ (B_\mathcal{F})_{2\_} \vee (A_\mathcal{F})_{\_3} \circ (B_\mathcal{F})_{3\_}$, which shows a $\bigvee$-decomposition of I into maximal rectangles. With our example, we have

$$\begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&1\\ 1&1&1&1&0\\ 1&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&1&0&0&0\\ 1&1&0&0&0\\ 1&1&0&0&0\\ 0&0&0&0&0 \end{pmatrix} \vee \begin{pmatrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 1&1&1&1&0\\ 0&0&0&0&0 \end{pmatrix} \vee \begin{pmatrix} 0&0&0&0&0\\ 1&0&0&0&1\\ 0&0&0&0&0\\ 1&0&0&0&1 \end{pmatrix}.$$

Remark 1. In factor analysis, factors considered to represent general categories, sometimes called concepts, of which the original variables are particular manifestations. The problem of interpretability of factors is a part of the whole process of factor analysis. From this point of view, interpretation of formal concepts as factors in BFA is clear for a user. Namely, the notion of a formal concept results as a mathematical formalization of the notion of a concept as worked out in the traditional logic. A formal concept $\langle A_l, B_l \rangle$ can be seen as a "unit of thought" consisting of a collection A_l of objects to which it applies (concept's extent) and a collection B_l of attributes to which it applies (concept's intent). Clear interpretability is one of the advantageous features of having formal concepts as factors.

Consider now the following problem we call the BFA Problem [4]:

INPUT: Boolean matrix I,
OUTPUT: smallest $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$ for which $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$.

As mentioned in [4], BFA Problem is reducible to the set covering optimization problem, for which we refer to [6]. Recall that in the set covering optimization problem we are given a set $\mathcal{U}$ and a collection $\mathcal{S} \subseteq 2^{\mathcal{U}}$ of subsets of $\mathcal{U}$ with $\bigcup \mathcal{S} = \mathcal{U}$. The goal is to find a set $\mathcal{C} \subseteq \mathcal{S}$ with the fewest sets (i.e. with $|\mathcal{C}|$ as small as possible) such that $\mathcal{C}$ covers $\mathcal{U}$, i.e. such that $\mathcal{U} = \bigcup \mathcal{C}$. The set covering optimization problem is a difficult problem. It is NP-hard and the corresponding decision problem is NP-complete. However, there exists an efficient greedy approximation algorithm for the set covering optimization problem which achieves an approximation ratio $\leq \ln(|\mathcal{U}|) + 1$, see [6].

The idea of the proof of Theorem 1 allows us to see that $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$ means that I is covered by the rectangles corresponding to $\langle A_l, B_l \rangle$'s from $\mathcal{F}$. Consequently, the BFA Problem is reducible to the set covering problem by putting $\mathcal{U} = \{\langle x_i, y_j \rangle \mid I_{ij} = 1\}$ and $\mathcal{S} = \{C \times D \mid \langle C, D \rangle \in \mathcal{B}(X, Y, I)\}$. That is, the set $\mathcal{U}$ to be covered is the set of all pairs for which the corresponding entry I_{ij} is 1, and the set $\mathcal{S}$ of sets which can be used for covering $\mathcal{U}$ is the set of "rectangular sets" of positions corresponding to formal concepts $\langle C, D \rangle \in \mathcal{B}(X, Y, I)$. The above-mentioned greedy approximation algorithm can therefore be used to find approximately optimal solutions for the BFA Problem. However, the particular nature of the BFA Problem enables us to speed up the algorithm. It is easy to see that if $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$, then $\mathcal{F}$ needs to contain all formal concepts from $\mathcal{B}(X, Y, I)$ which are both object concepts and attribute concepts, i.e. $\mathcal{O}(X, Y, I) \cap \mathcal{A}(X, Y, I) \subseteq \mathcal{F}$. Therefore, one can include $\mathcal{O}(X, Y, I) \cap \mathcal{A}(X, Y, I)$ in $\mathcal{F}$ right in the beginning. Recall that the set $\mathcal{O}(X, Y, I)$ of all object concepts and the set $\mathcal{A}(X, Y, I)$ of all attribute concepts is defined by

$$\mathcal{O}(X, Y, I) = \{\langle \{x\}^{\uparrow\downarrow}, \{x\}^{\uparrow} \rangle\} \text{ and } \mathcal{A}(X, Y, I) = \{\langle \{y\}^{\downarrow}, \{y\}^{\downarrow\uparrow} \rangle\}.$$

The resulting algorithm for computing the factors follows:

Algorithm 1 (Compute factor concepts)

INPUT: I (Boolean matrix)
OUTPUT: $\mathcal{F}$ (set of factor concepts)

```
set $\mathcal{S}$ to $\mathcal{B}(X, Y, I)$
set $\mathcal{U}$ to $\{\langle x_i, y_j \rangle \mid I_{ij} = 1\}$
set $\mathcal{F}$ to $\emptyset$
for each $\langle C, D \rangle \in \mathcal{S}$:
  if $(\langle C, D \rangle \in \mathcal{O}(X, Y, I) \cap \mathcal{A}(X, Y, I))$:
    add $\langle C, D\rangle$ to $\mathcal{F}$
    remove $\langle C, D\rangle$ from $\mathcal{S}$
    for each $\langle x, y\rangle \in C \times D$:
      remove $\langle x, y\rangle$ from $\mathcal{U}$
while ($\mathcal{U} \neq \emptyset$):
  do select $\langle C, D\rangle \in \mathcal{S}$ that maximizes $(C \times D) \cap \mathcal{U}$:
    add $\langle C, D\rangle$ to $\mathcal{F}$
    remove $\langle C, D\rangle$ from $\mathcal{S}$
    for each $\langle x, y\rangle \in C \times D$:
      remove $\langle x, y\rangle$ from $\mathcal{U}$
return $\mathcal{F}$
```

3 Experiments with Boolean factor analysis

In this section, we present experiments on factorization of Boolean matrices. In the experiments, we employed the algorithm described in the end of the previous section.

Experiment 1 The first experiment concerns analysis of factors which determine attributes of European Union countries. We have taken information from the Rank Order pages of the CIA World Factbook 2006[3] and created a Boolean matrix consisting of 27 rows (EU countries) and 141 columns (yes/no attributes). The attributes are scaled versions of the numerical values taken from the Factbook.

The total number of formal concepts present in the matrix is 3963. From this amount of concepts, Algorithm 1 computes only a small number of factor concepts. In particular, we obtained a set $\mathcal{F}$ of 49 factor concepts, i.e. formal concepts for which $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$. That is, the 27×141 matrix I has been decomposed into a Boolean product of a 27×49 Boolean matrix $A_{\mathcal{F}}$ representing a relationship between EU countries and the factors and a 49×141 Boolean matrix $B_{\mathcal{F}}$ representing a relationship between the factors and the original attributes. The factor concepts can be considered factors explaining completely the original 141 attributes. Note that the original attributes are socio-economic attributes. However, due to the limited scope, we restrict ourselves to listing the numbers of attributes and factors and leave the socio-economic interpretation of the factors to a future work, to be done possibly with an expert economist.

Experiment 2 We have used Algorithm 1 to compute factor concepts from large data sets. In the case of large data sets, it seems to be of interest whether there is a set of factors which approximately explain the data.

Here we present the results for the well-known MUSHROOM data set which can be found at the UCI Machine Learning Repository[4]. The MUSHROOM

[3] https://www.cia.gov/library/publications/download/
[4] http://www.ics.uci.edu/~mlearn/

database is presented as a collection of so-called “item sets”, i.e. it is a collection of sets of items. The collection can be transformed into a Boolean matrix with rows corresponding to items sets, attributes corresponding to items and table entries indicating whether an item set given by the row contains an item given by the column. The MUSHROOM database contains 8124 objects and 119 attributes. The corresponding Boolean matrix contains 238710 formal concepts.

Let us thus turn our attention to factor concepts which approximately explain the data. That is, our aim is to find a set $\mathcal{F}$ of factor concepts such that I is approximately equal to $A_{\mathcal{F}} \circ B_{\mathcal{F}}$. From the perspective of the results presented in this paper, solutions to the approximate factorization problem can be looked for by a slight modification of Algorithm 1. Recall that Algorithm 1 finishes its computation if each 1 in from the input table is covered by at least one factor. We might modify the halting condition of the algorithm so that

– it stops if the number of found factors exceeds threshold n; or
– it stops if the found factors cover “almost all 1s” present in the input matrix.

In either case, we obtain a set $\mathcal{F}$ of factor concepts so that $A_{\mathcal{F}} \circ B_{\mathcal{F}} \leq I$. It is desirable to have $A_{\mathcal{F}} \circ B_{\mathcal{F}}$ as close to I as possible while having a reasonable number of factors at the same time. The closeness of $A_{\mathcal{F}} \circ B_{\mathcal{F}}$ will be assessed as follows. For I and $\mathcal{F} \subseteq \mathcal{B}(X, Y, I)$, define $A(I, \mathcal{F})$ by

$$A(I, \mathcal{F}) = \frac{\text{Area}(\mathcal{F})}{\text{Area}(\mathcal{B}(X, Y, I))},$$

where

$$\text{Area}(\mathcal{G}) = |\{\langle i, j \rangle \mid (A_{\mathcal{G}} \circ B_{\mathcal{G}})_{ij} = 1\}|$$

for each $\mathcal{G} \subseteq \mathcal{B}(X, Y, I)$. Hence, Area$(\mathcal{G})$ is the number of 1s in the matrix given by a set of rectangles $\mathcal{G}$. As a consequence, Area$(\mathcal{B}(X, Y, I))$ is the number of 1s in the input matrix. $A(I, \mathcal{F})$ will be called a degree of approximation of I by $\mathcal{F}$. Furthermore, $A(I, \mathcal{F}) \cdot 100$ is the percentage of 1s in the input matrix I which are covered by factors from $\mathcal{F}$. Clearly, if $\mathcal{F}$ is a set of the exact factor concepts, i.e. $A_{\mathcal{F}} \circ B_{\mathcal{F}} = I$, then Area$(\mathcal{B}(X, Y, I))$ = Area$(\mathcal{F})$ which yields $A(I, \mathcal{F}) = 1$. Observe that $A(I, \mathcal{F}) \in [0, 1]$ and in addition, $A(I, \mathcal{F}) = 1$ iff I equals $A_{\mathcal{F}} \circ B_{\mathcal{F}}$, i.e., iff the factors completely explain the data.

Our experiments with the MUSHROOM data set have shown that most of the information contained in the data set can be expressed through a relatively small number of factor concepts. The results of our experiment can be depicted by a graph shown in Fig. 1. The graph shows a relationship between the number of factor concepts and the degree of approximation of the original data set. We can see from the picture that even if we take a relatively small number of factor concepts, we achieve high degree of approximation. For instance, if we take first 6 factor concepts returned by Algorithm 1, we get $F(I, \mathcal{F}) \cdot 100\% = 51.89\%$. This means that *more than half the information contained in the MUSHROOM data set can be explained by six factors.* The growth of the degree of approximation is rapid for first 10 factor concepts. The growth of the degree of approximation is shown in Table 1. The tables say that, for instance, if we wish to achieve 90.36%

Fig. 1. Relationship between the number of factors and the approximation of the original Boolean matrix.

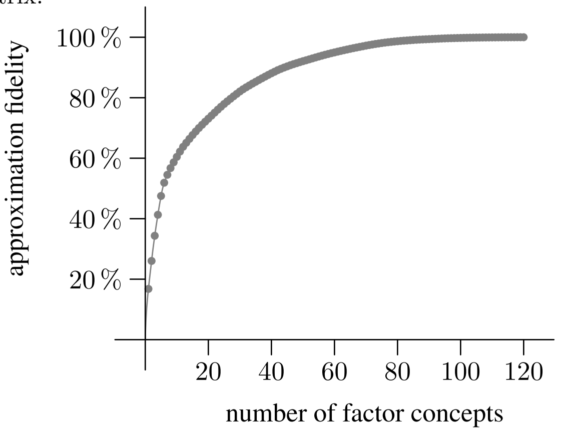


Table 1. Number of factor concepts vs. degree of approximation

factors (n)	1	2	3	4	5	6
fidelity (%)	16.78	26.03	34.35	41.29	47.51	51.89

factors (n)	10	20	45	60	100	119
fidelity (%)	60.44	73.07	90.36	94.98	99.76	100.00

approximation then it suffices to take 45 factor concepts which is significantly less than the number of the original attributes; 95% is guaranteed if we use 60 factor concepts, etc.

4 Conclusions and future research

We presented an algorithm for finding factors in Boolean factor analysis. The algorithm is based on a theorem, proven in our earlier paper, that the problem of BFA can be reduced to a problem of a covering of entries containing 1s in a given Boolean matrix I with maximal subrectangles of I which consist of 1s. This way, the problem of BFA is reducible to a particular instance of a set covering problem for which there exists an approximation algorithm. The algorithm can be sped up by further insight provided by formal concept analysis. We presented examples of Boolean factor analysis and experiments with the algorithm.

Future research will include the following problems:

– As we have seen, further insight provided by the particular nature of the set covering problem can speed up the greedy approximation algorithm. One such speed up results from the inclusion of mandatory concepts which we presented. Other ways of improving the algorithm as well as looking for other algorithms need to be investigated.

- Approximate decomposition, i.e. looking for A and B such that I is approximately equal to $A \circ B$, cf. the above experiments with MUSHROOM data. Both theoretical insight and experiments are needed in this direction.
- We did not impose any restrictions on $\mathcal{F}$ except for $I = A_{\mathcal{F}} \circ B_{\mathcal{F}}$. It might be desirable to look for $\mathcal{F}$ such that the number of attributes in the formal concepts' extents are restricted in one way or another. For instance, all formal concepts from $\mathcal{F}$ have approximately the same number of attributes, this means the level of generality of all factors is approximately the same. In general, a question of what is a good set $\mathcal{F}$ of factor concepts needs to be investigated. A small number of factor concepts, considered in this paper as the only criterion, might not always be the best one by itself.
- New factors can be seen as new attributes using which the objects are described. Since the number of new attributes (factors) is less than or equal to the number of the original attributes, a general question is this: Can we use the new attributes for more efficient reasoning and manipulation of objects? For instance: Is it useful to extract association rules over the objects which contain the new attributes? Is it useful to construct decision trees to classify objects using the new attributes?
- Another topic relates to the question of whether there are connections between formal concept analysis, concept lattices and related structures on the one hand, and associative neural networks on the other hand. In the light of [8] and the present paper, this question should be pursued. Useful hints in this respect could be provided by [2] and its followers such as [15].
- The last remark we want to make concerns the possibility to extend factorization of Boolean matrices to matrices containing more general entries, such as numbers from the unit interval $[0, 1]$, instead of just 0 and 1, expressing degrees to which attributes apply to objects. This is possible using an extension of formal concept analysis to the setting of fuzzy logic, see e.g. [3]. A paper on this topic is in preparation.

References

1. Bartholomew, D. J., Knott M.: *Latent Variable Models and Factor Analysis, 2nd Ed.*, London, Arnold, 1999.
2. Belohlavek R.: Representation of concept lattices by bidirectional associative memories. *Neural Computation* **12**, 10(2000), 2279–2290.
3. Belohlavek R.: *Fuzzy Relational Systems: Foundations and Principles.* Kluwer, Academic/Plenum Publishers, New York, 2002.
4. Belohlavek R., Vychodil V.: Formal concepts are optimal factors in Boolean factor analysis (submitted). Preliminary version appeared as On Boolean factor analysis with formal concepts as factors. SCIS & ISIS 2006, Int. Conf. Soft Computing and Intelligent Systems & Int. Symposium on Intelligent Systems, Sep 20-24, 2006, Tokyo, Japan, pp. 1054-1059.
5. Carpineto C., Romano G.: *Concept Data Analysis. Theory and Applications.* J. Wiley, 2004.
6. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: *Introduction to Algorithms, 2nd Ed.* MIT Press, 2001.

7. Cudeck R., MacCallum R. C. (Eds.): *Factor Analysis at 100: Historical Developments and Future Directions.* Lawrence Erlbaum Associates, Inc., 2007.
8. Frolov A. A., Húsek D., Muraviev I. P., Polyakov P. A.: Boolean factor analysis by Hopfield-like autoassociative memory. *IEEE Transactions on Neural Networks* Vol. **18**, No. 3, May 2007, pp. 698–707.
9. Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.
10. Harman H. H.: *Modern Factor Analysis, 2nd Ed.* The Univ. Chicago Press, Chicago, 1970.
11. Keprt A., Snášel V. Binary factor analysis with help of formal concepts. In *Proc. CLA 2004*, Ostrava, Czech Republic, 2004, pp. 90-101, ISBN 80–248–0597–9.
12. Keprt A., Snášel V.: Binary Factor Analysis with Genetic Algorithms. Proceedings of 4th IEEE WSTST 2005, Muroran, Japan. Springer Verlag, BerlinHeidelberg, Germany, 2005, pp. 1259-1268.
13. McDonald R. P.: *Factor Analysis and Related Methods.* Lawrence Erlbaum Associates, Inc., 1985.
14. Mickey, M.R., Mundle, P. and Engelman, L.: Boolean factor analysis. In: W.J. Dixon (Ed.), BMDP statistical software manual, vol. 2, 849–860, Berkeley, CA: University of California Press, 1990.
15. Rajapakse R.K, Denham M.: Fast access to concepts in concept lattices via bidirectional associative memories. *Neural Computation* **17**, 10(2005), 2291–2300.
16. Řezanková H., Húsek, D., Frolov, A.A.: Using standard statistical procedures for Boolean factorization. In *Proc. SIS 2003*, Naples, Italy, 2003, ISBN 88–8399–053–6.
17. Sirota, A. M., Frolov, A. A., Húsek, D.: Nonlinear factorization in sparsely encoded Hopfield-like neural networks. *ESANN European Symposium on Artifical Neural Networks*, Bruges, Belgium, 1999,pp. 387–392.
18. Spearman C.: General intelligence, objectively determined and measured. *Amer. J. Psychology* **15**(1904), 201–293.
19. Veiel H.O.: Psychopathology and Boolean factor analysis: a mismatch. *Psychological Medicine.* Vol. **15** (1985), issue 3 (Aug), pp. 623-630, ISSN 0033–2917.
20. Wille R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (Ed.): Ordered Sets, 445–470, Reidel, Dordrecht-Boston, 1982.

Further Galois Connections between Semimodules over Idempotent Semirings

Francisco J. Valverde-Albacete and Carmen Peláez-Moreno *

Dpto. de Teoría de la Señal y de las Comunicaciones.
Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés 28911. Spain
fva,carmen@tsc.uc3m.es

Abstract. In [14] a generalisation of Formal Concept Analysis was introduced with data mining applications in mind, $\mathcal{K}$-Formal Concept Analysis, where incidences take values in certain kinds of semirings, instead of the standard Boolean carrier set. The construction leading to the pair of dually (order) isomorphic lattices can be further manipulated to obtain the three other types of Galois Connections providing a fuller set of tools to interpret any relations between data. We relate this result to previous descriptions of certain instances of such Galois Connections in qualitative data analysis and provide concrete examples of them related to $\overline{\mathbb{R}}_{\max,+}$-semimodules in quantitative data analysis.

1 Motivation: Lattices related to an Incidence

Data analysis results improve when many different tools are offered to the practitioner. Consider then the *modal operators* ([13], def. 3.8.2; [6]) introduced by a Boolean matrix, $I \in 2^{G\times M}$, over a set of objects, $A \in 2^G$ and, dually, over sets $B \in 2^M$ of attributes operated by the converse relation $I^{\mathrm{t}} \in 2^{M\times G}$ as listed in Table 1. Formal Concept Analysis adepts may recognise the extent and intent polars in the sufficiency operators for a relation, $[[I\,]]\,(A) = A'$, $[[I^{\mathrm{t}}]]\,(B) = B'$, but also their closure operators, $[[I^{\mathrm{t}}]][[I\,]](A) = A''$, $[[I\,]]\,[[I^{\mathrm{t}}]]\,(B) = B''$.

Perhaps less known is that the pairs of operators in the first and second rows of Table 1 define the *neighbourhood lattices*: For a formal context (G, M, I) define the *span* of a set of objects as: $\mathrm{span}(A) := \langle I\rangle(A) = (A)^{I}_{\exists}$. This is the set of attributes related to some $g \in A$ [1]. Similarly, define for its dual context (M, G, I^{t}) the *content* of a set of attributes, $\mathrm{content}(B) = [I^{\mathrm{t}}]\,(B) = (B)^{\forall}_{I}$, as the set of objects which can be completely described by the attributes in B . Next consider the set $\mathfrak{N}(G, M, I)$ (for Ger. *Nachbar*, neighbour) of *neighbour pairs*, $(A, B) \in \mathfrak{N}(G, M, I)$, such that $\mathrm{span}(A) = (A)^{I}_{\exists} = B \Leftrightarrow A = (B)^{\forall}_{I} = \mathrm{content}(B)$. Then we can state the:

* This work has been partially supported by a grants from the Spanish Government-Comisión Interministerial de Ciencia y Tecnología project TEC2005-04264/TCM.

[1] The second, operator notation is closer to Galois connection theory as explained below and relates better to normal notation in Formal Concept Analysis.

Table 1. Modal operators over a relation and its converse for sets of objects $A \subseteq G$ and attributes $B \subseteq M$. The misalignment in the first two rows is intentional.

possibility operator over G $\langle I \rangle (A) = \{ m \in M \mid (\exists g \in G)[g \in A \wedge gIm] \}$	necessity operator over M $[I^{t}] (B) = \{ g \in G \mid (\forall m \in M)[mI^{t}g \Rightarrow m \in B) \}$
necessity operator over G $[I] (A) = \{ m \in M \mid (\forall g \in G)[gIm \Rightarrow g \in A) \}$	possibility operator over M $\langle I^{t} \rangle (B) = \{ g \in G \mid (\exists m \in M)[m \in B \wedge mI^{t}g] \}$
sufficiency operator over G $[[I]] (A) = \{ m \in M \mid (\forall g \in G)[g \in A \Rightarrow gIm) \}$	sufficiency operator over M $[[I^{t}]] (B) = \{ g \in G \mid (\forall m \in M)[m \in B \Rightarrow mI^{t}g) \}$
dual sufficiency operator over G $\langle\langle I \rangle\rangle (A) = \{ m \in M \mid (\exists g \in G)[g \notin A \wedge g \not{I} m] \}$	dual sufficiency operator over M $\langle\langle I^{t} \rangle\rangle (B) = \{ g \in G \mid (\exists m \in M)[m \notin B \wedge m \not{I}^{t} g] \}$

Theorem 1 (Fundamental theorem of Neighbourhood lattices [6]). *The neighbourhood lattice, $\mathfrak{N}(G, M, I)$, is a complete lattice in which infimum and supremum are given by:*

$$\bigwedge_{t \in T} (A_t, B_t) = \left(\bigcap_{t \in T} A_t, \left(\left(\bigcap_{t \in T} B_t \right)^{\forall}_{I} \right)^{I}_{\exists} \right) \bigvee_{t \in T} (A_t, B_t) = \left(\left(\left(\bigcup A_t \right)^{I}_{\exists} \right)^{\forall}_{I}, \bigcup_{t \in T} B_t \right)$$

Conversely, a complete lattice V is isomorphic to $\mathfrak{N}(G, M, I)$ if and only if there are mappings[2] *$\sigma : G \to V$ and $\kappa : M \to V$ such that $\sigma(G) \cup \{0\}$ is join-dense in V and $\kappa(M) \cup \{1\}$ is meet-dense in V and gIm is equivalent to $\sigma(g) \nleq \kappa(m)$ for all $g \in G$ and all $m \in M$.*

This result shows that $\mathfrak{N}(G, M, I)$ is isomorphic to $\underline{\mathfrak{B}}(G, M, \not{I})$ [6], but the drawing and interpretation in terms of neighbourhood pairs must differ, since the systems of spans and contents are now order isomorphic: if (A_1, B_1), (A_2, B_2) are neighbour pairs, they are ordered by the relation: $(A_1, B_1) \leq (A_2, B_2) \iff A_1 \leq A_2 \iff B_1 \leq B_2$.

In a different train of thought and with algebraic applications in mind, when considering finite incidences to be represented by Boolean matrices, $I = [I_{ij}]$ and sets of objets by Boolean (column) vectors, we may write: $\langle I \rangle (A) = \bigvee_j I_{ij} \wedge a_j$, $[I] (A) = \bigwedge_j I_{ij} \to a_j$, $\langle\langle I \rangle\rangle (A) = \bigvee_j \overline{I_{ij}} \wedge \overline{x_j}$ and $[[I]] (A) = \bigwedge_j a_j \to I_{ij}$, where $\to$ indicates the logical conditional. It is clear that definition of the possibility operator can be understood in terms of matrix multiplication in the Boolean algebra where $\vee$ is addition and $\wedge$ multiplication, so that we could write for proper finite sets of objects and attributes with cardinals $|G| = g, |M| = m$,

[2] Our names for the neighbour pair-creating functions to avoid those already taken by Formal Concept Analysis.

an incidence $I \in 2^{g \times m}$ and a set of objects $A \in 2^{g \times 1}$, $\langle I \rangle (A) = I^{\mathrm{t}} \cdot A$ with the conventional multiplication betwen boolean matrices. But that is not at all apparent in the case of the other operators.

The question arises when considering a particular incidence whether there are any more lattices related to it, and if so what their properties are. For instance, are there analogues of the constructions and methods in Formal Concept Analysis for the composition of the other operators? Are there similar operators on typical domains for data mining like $\mathbb{N}_0$ or $\mathbb{R}_0^+$? Do these operators admit a matrix representation?

In this paper we try to answer affirmatively to the questions posed above, providing analogues for the three other types of lattice stemming from incidences taking values in reflexive semifields. For that purpose apart from the wider scope of Galois connections between arbitrary orders in section 2, we reformulate in section 3 the construction of Galois connections in idempotent semimodules which are idempotent analogues of vector spaces and provide examples and a linear algebraic setting for concrete instances of these in section 4.

2 Galois Connections and Adjunctions

Let $\mathcal{P} = \langle P, \leq_{\mathcal{P}} \rangle$ and $\mathcal{Q} = \langle Q, \leq_{\mathcal{Q}} \rangle$ be partially ordered sets. We introduce the following naming conventions for the purpose of clarification [3] :

Definition 1. *1.* (λ, ρ) *is a* TYPE OO *Galois connection or* (Galois) adjunction (on the left)*, and write* $(\lambda, \rho) : \mathcal{P} \leftrightharpoons \mathcal{Q}$ *iif:* $\forall p \in P, q \in Q \quad \lambda(p) \leq_{\mathcal{Q}} q \leftrightharpoons p \leq_{\mathcal{P}} \rho(q)$ *, that is, the functions are covariant, and we say that* λ *is the* lower or left adjoint *while* ρ *is the* upper or right adjoint .

2. $(\rho, \lambda) : \mathcal{P} \rightleftharpoons \mathcal{Q}$ *is a* TYPE II *Galois connection or* (Galois) adjunction (on the right) *iff:* $\forall p \in P, q \in Q \quad \rho(p) \geq_{\mathcal{Q}} q \Leftrightarrow p \leq_{\mathcal{P}} \lambda(q)$ *, both functions are covariant,* ρ *is the upper adjoint, and* λ *the lower adjoint.*

3. (φ, ψ) *is a* TYPE OI *Galois connection, or* Galois Connection proper*, and write* $(\varphi, \psi) : \mathcal{P} \rightharpoonup \mathcal{Q}$ *iff:* $\forall p \in P, q \in Q \quad \varphi(p) \geq_{\mathcal{Q}} q \Leftrightarrow p \leq_{\mathcal{P}} \psi(q)$ *, that is, both functions are contravariant. For that reason they are sometimes named* contravariant or symmetric adjunctions on the right*. Note that* (ψ, φ) *is also a* TYPE OI *Galois connection.*

4. $(\triangle, \triangle')$ *is a* TYPE IO*, or* co-Galois connection*, and write* $(\triangle, \triangle') : \mathcal{P} \leftharpoondown \mathcal{Q}$ *if:* $\forall p \in P, q \in Q \quad \triangle(p) \leq_{\mathcal{Q}} q \Leftrightarrow p \geq_{\mathcal{P}} \triangle'(q)$ *, that is, both functions are contravariant. For that reason they are sometimes named* contravariant or symmetric adjunctions on the left. $(\triangle', \triangle)$ *is also a co-Galois connection.*

Our classification of Galois connections stresses the compositions with order- and dual order-isomorphisms, or *anti-isomorphisms*. We take the TYPE OO Galois connection to be a basic adjunction composed with an even number of

[3] For a revision of the genesis and importance of Galois Connections and adjunctions see [3], as well as a discussion of the different notation and nomenclatures for these concepts. See [4] for an early tutorial with mathematical applications in mind.

anti-isomorphism for the domain or range orders. Consequently, a TYPE II Galois connection, is a basic adjunction with an odd number of anti-isomorphisms composed on both the domain and range orders. To obtain contravariance we compose with an odd number of anti-isomorphism on the ranges to obtain a TYPE OI Galois connection. Finally, to obtain a co-Galois connection, we compose with an odd number of anti-isomorphisms only on the domain, i.e. to get a a TYPE IO Galois connection.

Table 2 summarises briefly the main properties of all types of Galois connections. Furthermore, as a sort of graphical summary, we introduce the diagram to

Table 2. Summary of Galois connections and their properties, for P, Q posets.

Left Adjunction: $(\lambda, \rho) : \mathcal{P} \leftrightharpoons \mathcal{Q}$	Galois connection: $(\varphi, \psi) : \mathcal{P} \leftharpoonup\!\rightharpoonup \mathcal{Q}$
$\forall p \in P, q \in Q \quad \lambda(p) \leq_{\mathcal{Q}} q \leftrightharpoons p \leq_{\mathcal{P}} \rho(q)$	$\forall p \in P, q \in Q \quad \varphi(p) \geq_{\mathcal{Q}} q \Leftrightarrow p \leq_{\mathcal{P}} \psi(q)$
$I_{\mathcal{P}} \leq \rho \circ \lambda$ and $I_{\mathcal{Q}} \geq \lambda \circ \rho$	$I_{\mathcal{P}} \leq \psi \circ \varphi$ and $I_{\mathcal{Q}} \leq \varphi \circ \psi$
$\lambda = \lambda \circ \rho \circ \lambda$ and $\rho = \rho \circ \lambda \circ \rho$	$\varphi = \varphi \circ \psi \circ \varphi$ and $\psi = \psi \circ \varphi \circ \psi$
λ monotone, residuated	φ antitone
ρ monotone, residual	ψ antitone
λ join-preserving, ρ meet-preserving	φ join-inverting, ψ join-inverting
co-Galois connection: $(\triangle, \triangle') : \mathcal{P} \leftharpoondown\!\rightharpoondown \mathcal{Q}$	Right Adjunction: $(\rho, \lambda) : \mathcal{P} \rightleftharpoons \mathcal{Q}$
$\forall p \in P, q \in Q \quad \triangle(p) \leq_{\mathcal{Q}} q \Leftrightarrow p \geq_{\mathcal{P}} \triangle'(q)$	$\forall p \in P, q \in Q \quad \rho(p) \geq_{\mathcal{Q}} q \Leftrightarrow p \leq_{\mathcal{P}} \lambda(q)$
$I_{\mathcal{P}} \geq \triangle' \circ \triangle$ and $I_{\mathcal{Q}} \geq \triangle \circ \triangle'$	$I_{\mathcal{P}} \geq \lambda \circ \rho$ and $I_{\mathcal{Q}} \leq \rho \circ \lambda$
$\triangle = \triangle \circ \triangle' \circ \triangle$ and $\triangle' = \triangle' \circ \triangle \circ \triangle'$	$\rho = \rho \circ \lambda \circ \rho$ and $\lambda = \lambda \circ \rho \circ \lambda$
$\triangle$ antitone	ρ monotone, residual
$\triangle$ antitone	λ monotone, residuated
$\triangle$ meet-inverting, $\triangle'$ meet-inverting	ρ meet-preserving, λ join-preserving

the upper left-hand corner of Figure 1 as the pattern that carries the structures described in ([3], §1.2) and llustrated at the top left of Figure 1:

- A closure system, $\rho(Q) = \overline{P}$, the closure range of the right adjoint (see below).
- An interior system, $\lambda(P) = \underline{Q}$, the kernel range of the left adjoint (see below).
- A closure function (also "closure operator" [6,2]) $\gamma_{\mathcal{P}} = \rho \circ \lambda \geq_{\mathcal{P}} I_{\mathcal{P}}$, from P to the closure range $\rho(Q)$, with adjoint inclusion map $\hookrightarrow_{\mathcal{P}}$, where $I_{\mathcal{P}}$ denotes the identity over P.
- A kernel function (also "interior operator" [6], "kernel operator") $\kappa_{\mathcal{P}} = \lambda \circ \rho \leq_{\mathcal{Q}} I_{\mathcal{Q}}$, from Q to the range of $\lambda(P)$, with adjoint inclusion map $\hookrightarrow_{\mathcal{Q}}$, where $I_{\mathcal{Q}}$ denotes the identity over Q.
- a *perfect adjunction* $(\tilde{\lambda}, \tilde{\rho}) : \overline{P} \leftrightharpoons \underline{Q}$, that is, an order isomorphism between the closure and kernel ranges $\overline{P}$ and $\underline{Q}$.

However, a Galois connection proper can be seen in the top right of Figure 1) whose ranges are *both* closure systems and *both* compositions closure operators

$$\begin{tikzcd}[column sep=huge, row sep=huge]
P \arrow[r, shift right, "\lambda"'] \arrow[d, shift right, "\gamma_P"'] \arrow[dr, "\lambda"'] & Q \arrow[l, shift right, "\varrho"'] \arrow[d, shift right, "\kappa_Q"'] \arrow[dl, "\varrho"] \\
\varrho(Q)=\overline{P} \arrow[u, shift right, "\hookrightarrow_P"'] \arrow[r, shift right, "\tilde{\lambda}"'] & \lambda(P)=\underline{Q} \arrow[u, shift right, "\hookrightarrow_Q"'] \arrow[l, shift right, "\tilde{\varrho}"']
\end{tikzcd}$$

Type OO:
Left Adjunction, $(\lambda, \varrho) : \mathcal{P} \leftrightarrows \mathcal{Q}$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
P \arrow[r, shift right, "\varphi"'] \arrow[d, shift right, "\gamma_P"'] \arrow[dr, "\varphi"'] & Q \arrow[l, shift right, "\psi"'] \arrow[d, shift right, "\gamma_Q"'] \arrow[dl, "\psi"] \\
\psi(Q)=\overline{P} \arrow[u, shift right, "\hookrightarrow_P"'] \arrow[r, shift right, "\tilde{\varphi}"'] & \varphi(P)=\overline{Q} \arrow[u, shift right, "\hookrightarrow_Q"'] \arrow[l, shift right, "\tilde{\psi}"']
\end{tikzcd}$$

Type OI:
Galois connection: $(\varphi, \psi) : \mathcal{P} \rightharpoonup\!\!\leftharpoondown \mathcal{Q}$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
P \arrow[r, shift right, "\triangle"'] \arrow[d, shift right, "\kappa_P"'] \arrow[dr, "\triangle"'] & Q \arrow[l, shift right, "\triangle'"'] \arrow[d, shift right, "\kappa_Q"'] \arrow[dl, "\triangle'"] \\
\triangle'(Q)=\underline{P} \arrow[u, shift right, "\hookrightarrow_P"'] \arrow[r, shift right, "\tilde{\triangle}"'] & \triangle(P)=\underline{Q} \arrow[u, shift right, "\hookrightarrow_Q"'] \arrow[l, shift right, "\tilde{\triangle}'"']
\end{tikzcd}$$

Type IO:
Co-Galois connection, $(\triangle, \triangle') : \mathcal{P} \leftharpoondown\!\!\rightharpoondown \mathcal{Q}$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
P \arrow[r, shift left, "\varrho"] \arrow[d, shift right, "\kappa_P"'] \arrow[dr, "\varrho"'] & Q \arrow[l, shift left, "\lambda"] \arrow[d, shift right, "\gamma_Q"'] \arrow[dl, "\lambda"] \\
\lambda(Q)=\underline{P} \arrow[u, shift right, "\hookrightarrow_P"'] \arrow[r, shift left, "\tilde{\varrho}"] & \varrho(P)=\overline{Q} \arrow[u, shift right, "\hookrightarrow_Q"'] \arrow[l, shift left, "\tilde{\lambda}"]
\end{tikzcd}$$

Type II:
Right Adjunction, $(\varrho, \lambda) : \mathcal{P} \rightleftarrows \mathcal{Q}$

Fig. 1. Diagrams visually depicting the maps and structures involved in the adjunction on the left $(\lambda, \varrho) : P \leftrightarrows Q$ (top left), Galois connection $(\varphi, \psi) : P \rightharpoonup\!\!\leftharpoondown Q$ (top right), the co-Galois connection $(\triangle, \triangle') : P \leftharpoondown\!\!\rightharpoondown Q$ (bottom left) and the adjunction on the right $(\varrho, \lambda) : P \rightleftarrows Q$ (bottom right) between two partially ordered sets (adapted from [3,13]). Closure operators are denoted by γ_P, γ_Q , interior (kernel) operators by κ_P, κ_Q , closure systems by $\overline{P}, \overline{Q}$ and interior (kernel) systems by $\underline{P}, \underline{Q}$.

due to the dualisation of the second set (we write γ_Q for the new closure operator), resulting in the well-known *perfect Galois connection*, $(\tilde{\varphi}, \tilde{\psi}) : \overline{\mathcal{P}} \rightharpoonup\!\!\leftharpoondown \overline{\mathcal{Q}}$, a pair of *dual* order isomorphism between closure ranges lying at the heart of Formal Concept Analysis. The diagrams in the bottom left and right show analogue structures for co-Galois connections and right adjunctions respectively.

As an example of all the above, consider $P = 2^G, Q = 2^M$ the powersets of a set of objects G and a set of attributes M . Then for each relation $R \in 2^{G \times M}$ we have (adapted from [4]):

– a Galois connection (TYPE OI) $(\cdot\ ^R, \cdot\ ^R) : 2^G \rightharpoonup\!\!\leftharpoondown 2^M$, with dually isomorphic (closure) lattices of object and attribute sets at the heart of Formal Concept Analysis.
– a left adjunction (TYPE OO) $(\cdot\ ^R_\exists, \cdot\ ^\forall_R) : 2^G \leftrightarrows 2^M$, with closure system $\left(2^M\right)^\forall_R$ that we call the *neighbourhood lattice of objects*.
– a right adjunction (TYPE II) $(\cdot\ ^R_\forall, \cdot\ ^\exists_R) : 2^G \rightleftarrows 2^M$, with closure system $\left(2^G\right)^R_\forall$ that we call the *neighbourhood lattice of attributes*.

– a co-Galois connection (TYPE IO) $(\cdot \overset{R}{-}, \cdot \underset{R}{-}) : 2^G \leftrightarrows 2^M$, with dually isomorphic (kernel) lattices of object and attribute sets.

3 Galois Connections between Idempotent Semimodules

It is not straightforward to describe the examples in section 2 in the algebra of Boolean matrices. For this purpose, we develop the more encompassing the concept of a Galois connection between two idempotent semimodules next.

3.1 Idempotent Semirings and Semifields

Basic definitions. A *semiring* $\mathcal{S} = \langle S, \oplus, \otimes, \epsilon, e \rangle$ is a structure where the additive structure, $\langle S, \oplus, \epsilon \rangle$, is a commutative monoid and the multiplicative structure, $\langle S \backslash \{\epsilon\}, \otimes, e \rangle$, a monoid whose multiplication distributes over addition from right and left and whose neutral element is absorbing for $\otimes$, $\forall x \in K,\ \epsilon \otimes x = \epsilon$. On any semiring $\mathcal{S}$ left and right multiplications can be defined: $\mathrm{L}_a : S \to S, b \mapsto \mathrm{L}_a(b) = ab$, and $\mathrm{R}_a : S \to S, b \mapsto \mathrm{R}_a(b) = ba$. A *commutative semiring* is a one whose multiplicative structure is commutative.

For instance, let $\mathcal{S} = \langle S, \oplus, \otimes, \epsilon, e \rangle$ be a semiring. Then the semiring of (square) matrices over $\mathcal{S}$ is $\mathcal{M}_n(\mathcal{S}) = \langle S^{n \times n}, \oplus, \otimes, \mathcal{E}, E \rangle$, with $S^{n \times n}$ denoting the set of square matrices over the semiring with matrix operations: $(A \oplus B)_{ij} = A_{ij} \oplus B_{ij}, 0 \leq i, j \leq n$ and $(A \otimes B)_{ij} = \bigoplus_{k=1}^{n} A_{ik} \otimes B_{kj}, 0 \leq i, j \leq n$, null element the matrix $\mathcal{E}$, $\mathcal{E}_{ij} = \epsilon, 0 \leq i, j \leq n$ and unit E, $E_{ii} = e, 0 \leq i \leq n$, $E_{ij} = \epsilon, 0 \leq i, j \leq n, i \neq j$. Such semirings are not commutative in general even if $\mathcal{S}$ is, except for $\mathcal{M}_1(\mathcal{S}) = \mathcal{S}$.

A *semifield* is a semiring whose multiplicative structure $\langle S \backslash \{\epsilon\}, \otimes \rangle$ is a group, that is, there is an operation, $\cdot^{-1} : S \backslash \{\epsilon\} \to S \backslash \{\epsilon\}$ such that $\forall a \in S, a \otimes a^{-1} = a^{-1} \otimes a = e_{\mathcal{S}}$. For commutative semifields, whose multiplicative structure is a commutative group, we have $(a \otimes b)^{-1} = a^{-1} \otimes b^{-1}$.

An *idempotent semiring or dioid* (for double monoid), $\mathcal{D}$, is a semiring whose addition is idempotent, $\forall a \in D, a \oplus a = a$, that is, whose additive structure $\langle D, \oplus, \epsilon \rangle$ is an *idempotent semigroup* . Compared to a ring, an idempotent semiring crucially lacks additive inverses. All idempotent commutative monoids $\langle D, \oplus, \epsilon \rangle$ are endowed with a *natural order* $\forall a, b \in D, a \preceq b \iff a \oplus b = b$, which turns them into $\vee$-semilattices with least upper bound defined as $a \vee b = a \oplus b$. Moreover, the neutral element for the additive structure of semiring $\mathcal{D}$ is the infimum for this natural order, $\epsilon_{\mathcal{D}} = \bot_{\mathcal{D}}$. Hence all dioids are sup-semilattices $\langle D, \preceq \rangle$ with a bottom element.

A dioid whose multiplicative structure is a group is an *idempotent semifield.* The formula for the infimum in such case was already put forward by Dedekind [3]: the meet law is: $a \wedge b = a^{-1} \otimes (a \oplus b) \otimes b^{-1}$, hence idempotent semifields are already lattices. In this paper, we focus on two idempotent semifields[4]:

[4] We use $:=$ (read "becomes") to pass from abstract to concrete algebra.

1. The *maxplus semifield*, $\mathbb{R}_{\max,+} = \langle \mathbb{R} \cup \{ -\infty \}, \max, +, -\infty, 0 \rangle$ with inverse $\cdot^{-1} := -\cdot$ is an idempotent commutative semifield.
2. The *minplus semifield*, $\mathbb{R}_{\min,+} = \langle \mathbb{R} \cup \{ \infty \}, \min, +, \infty, 0 \rangle$ is an idempotent commutative semifield, with the same inverse as the previous example.

Complete Semirings and Dioids. A semiring $\mathcal{S}$ is *complete*, if for any index set I including the empty set, and any $\{a_i\}_{i\in I} \subseteq \mathcal{S}$ the (possibly infinite) summations $\bigoplus_{i\in I} a_i$ are defined and the distributivity conditions: $\left(\bigoplus_{i\in I} a_i\right) \otimes c = \bigoplus_{i\in I} (a_i \otimes c)$ and $c \otimes \left(\bigoplus_{i\in I} a_i\right) = \bigoplus_{i\in I} (c \otimes a_i)$, are satisfied. Note that for $c = e$ the above demand that infinite sums have a result. A dioid $\mathcal{D}$ is *complete*, if it is complete as a naturally ordered set $\langle D, \preceq \rangle$ and left (L_a) and right (R_a) multiplications are lower semicontinuous, that is, residuated: $\forall a, b \in D, a \preceq b, \forall c \in D, \mathrm{L}_c(a) \preceq \mathrm{L}_c(b), \mathrm{R}_c(a) \preceq \mathrm{R}_c(b)$. This implies that infinite sums are defined in terms of suprema: $\bigoplus_{i\in I} a_i = \sup^{\mathcal{D}}_{i\in I}\{a_i\}$, $\forall \{a_i\}_{i\in I} \subseteq \mathcal{D}$, with the convention that $\bigoplus_{i\in\varnothing} a_i = \epsilon$. A less strict definition is: a dioid $\mathcal{D}$ is said to be *boundedly complete*[5] if every set $\mathcal{M} \subseteq \mathcal{D}$ order-bounded from above has a least upper bound $\sup \mathcal{M} \in \mathcal{D}$.

In a complete idempotent semiring, $\mathcal{D}$, because it is a sup-semilattice $\langle D, \vee \rangle$ with bottom element ϵ, the infima of subsets also exist, hence $\mathcal{D}$ is in fact a *complete lattice* . Further, for a complete idempotent semifield, this infimum can be computed from the supremum as: $\forall a, b \in D, a \wedge b = (a^{-1} \vee b^{-1})^{-1}$ and multiplication also distributes with respect to this infimum. For instance, the Boolean semiring, $\mathcal{B} = \langle \mathbb{B}, \vee, \wedge, 0, 1 \rangle$, with $\mathbb{B} = \{0, 1\}$ is complete, idempotent and commutative.

A complete idempotent semiring $\mathcal{D}$ can never be a semifield unless it is isomorphic to the Boolean semifield $\mathcal{B}$. For instance, its maximal element $\top_{\mathcal{D}}$ satisfies $\top_{\mathcal{D}} \otimes \top_{\mathcal{D}} = \top_{\mathcal{D}}$, hence it cannot have an inverse. $\mathbb{R}_{\max,+}$ is incomplete because its bottom has no inverse in the sense that $\infty + (-\infty) = -\infty \neq 0$. For a similar reason, $\mathbb{R}_{\min,+}$ is incomplete: its bottom has no inverse: $-\infty + \infty = \infty \neq 0$. (The apparent incongruence between these sums is about to be solved.)

The Completion of Idempotent Semifields. Let a *lattice-ordered group* $\mathcal{G} = \langle G, \preceq, \otimes \rangle$ be a lattice $\langle G, \preceq \rangle$ endowed with a group operation such that the multiplications on either side are isotone (or lower semicontinuous):

$$a, b, c \in G, a \preceq b \Rightarrow c \otimes a \preceq c \otimes b, a \otimes c \preceq b \otimes c$$

A lattice-ordered group $\mathcal{G}$ is said to be *conditionally complete* if it is conditionally complete as a lattice. Every conditionally-complete lattice-ordered group is commutative in the product operation. Also, lattice-ordered semigroups which are not singletons have no least, nor greatest elements. For instance, dioids are lattice-ordered semigroups for the natural order, hence they are commutative in the product and incomplete, lacking bottom $\perp$ or top $\top$ elements, or both. However, we may complete a lattice ordered group with the *canonical enlargement construction* as follows:

[5] Also *conditionally complete* or simply *complete* in the context of dioids [5].

Construction 2 (([9,10,11,12]) Canonical enlargement of a lattice-ordered group). *For any lattice-ordered group $\mathcal{G} = \langle G, \preceq, \otimes \rangle$: adjoin two elements $\bot$ and $\top$ to G to obtain $\overline{G} = G \cup \{\bot, \top\}$ and extend the order to $\overline{G}$ as $\bot \preceq a \preceq \top, \forall a \in \bar{G}$. Then extend the product to two different operations,* upper, $\dot{\otimes}$, and lower, $\underset{\cdot}{\otimes}$, multiplications*:*

$$a \underset{\cdot}{\otimes} b = \begin{cases} \bot & \textit{if } a, b \in G \cup \{\bot, \top\}, \textit{with } a = \bot, \ \textit{or } b = \bot; \\ \top & \textit{if } a, b \in G \cup \{\top\}, \textit{with } a = \top, \ \textit{or } b = \top; \\ a \otimes b & \textit{if } a, b \in G; \end{cases} \tag{1}$$

$$a \dot{\otimes} b = \begin{cases} \top & \textit{if } a, b \in G \cup \{\bot, \top\}, \textit{with } a = \top, \ \textit{or } b = \top; \\ \bot & \textit{if } a, b \in G \cup \{\bot\}, \textit{with } a = \bot, \ \textit{or } b = \bot; \\ a \otimes b & \textit{if } a, b \in G; \end{cases} \tag{2}$$

to obtain the structure $\overline{\mathcal{G}} = \langle \overline{G}, \preceq, \dot{\otimes}, \underset{\cdot}{\otimes} \rangle$, known as the canonical enlargement of $\mathcal{G} = \langle G, \preceq, \otimes \rangle$ *. In this structure, $\underset{\cdot}{\otimes}$ and $\dot{\otimes}$ are associative and commutative over $\overline{G}$, as the original $\otimes$ was over G , and the isotony of the product with respect to the natural order extends to $\overline{\mathcal{G}}$. Furthermore, if e is the unit element of $\langle G, \otimes \rangle$, it is similarly the unit of $\langle \overline{G}, \dot{\otimes} \rangle$ and $\langle \overline{G}, \underset{\cdot}{\otimes} \rangle$.*

This is the basis for the completion of idempotent semifields, to follow:

Construction 3. *The* top completion *[5] of a dioid $\mathcal{D}$ is another dioid $\overline{\mathcal{D}} = \langle \overline{D}, \underset{\cdot}{\oplus}, \underset{\cdot}{\otimes}, \epsilon, e \rangle$ where: $\overline{D} = D \cup \{\top\}$ and in which $\underset{\cdot}{\otimes}$ coincides with its definition in construction 2 when $\mathcal{D}$ is considered as bearing a lattice-ordered (multiplicative semi-)group, and we extend $\oplus$ with the extra top-element:*

$$a \underset{\cdot}{\oplus} b = \begin{cases} \top & \textit{if } a = \top \textit{ or } b = \top; \\ a \oplus b, & \textit{if } a, b \in D; \end{cases} \tag{3}$$

Construction 4 (Top Completion of an idempotent semifield). *Given an (incomplete) idempotent semifield $\mathcal{D}$, on its top enlargement as a dioid by construction 3, $\overline{\mathcal{D}}$, we extend the* notation *for the inverse with the following conventions: $\epsilon^{-1} = \top, \top^{-1} = \epsilon$. In that way we have two related complete idempotent semifield structures:*

- *a complete lattice for the natural order $\langle \overline{D}, \preceq \rangle$, the one we have been focusing on, $\overline{\mathcal{D}} = \langle \overline{D}, \underset{\cdot}{\oplus} = \vee, \underset{\cdot}{\otimes}, \bot, e \rangle$, and*
- *a complete lattice for the* dual of the natural order, $\langle \overline{D}, \preceq^d \rangle = \langle \overline{D}, \succeq \rangle$ $\overline{\mathcal{D}}^d = \langle \overline{D}, \dot{\oplus} = \wedge, \dot{\otimes}, \top, e \rangle$ *where the meet is defined (on $\mathcal{D}$) as above and the definition of $\dot{\otimes}$ follows that in construction 2.*

Using constructions 2, 3 and 4, already invoked by Moreau [9] we have:

- The top completion of $\mathbb{R}_{\max,+}$ is $\overline{\mathbb{R}}_{\max,+} = \langle \mathbb{R} \cup \{-\infty\, \infty\}, \max, \underset{\cdot}{+}, -\infty, 0 \rangle$, the *completed Maxplus semifield.*

– The top completion of $\mathbb{R}_{\min,+}$ is $\overline{\mathbb{R}}_{\min,+} = \langle \mathbb{R} \cup \{-\infty, \infty\}, \min, \dot{+}, \infty, 0\rangle$ the *completed Minplus semifield* .

Note that in this notation we have $-\infty \underset{\cdot}{+} \infty = -\infty$ and $-\infty \dot{+} \infty = \infty$, which solves several issues in dealing with the separately completed dioids, as promised.

In the completed structure, we have the following De Morgan-like relations between the multiplications, their residuals and inversion:

Property 5 ([12], lemma 2.2). *In the top enlargement $\overline{\mathcal{S}}$ of any commutative semifield $\mathcal{S}$ we have:*

$$\begin{aligned} (a \underset{\cdot}{\oplus} b)^{-1} &= a^{-1} \dot{\oplus} b^{-1} & \qquad (a \dot{\oplus} b)^{-1} &= a^{-1} \underset{\cdot}{\oplus} b^{-1} \\ (a \underset{\cdot}{\otimes} b)^{-1} &= a^{-1} \dot{\otimes} b^{-1} & \qquad (a \dot{\otimes} b)^{-1} &= a^{-1} \underset{\cdot}{\otimes} b^{-1} \end{aligned} \tag{4}$$

Furthermore if $\overline{\mathcal{S}}$ is idempotent, the residuals

$$a \underset{\cdot}{\otimes} b \preceq c \Leftrightarrow b \preceq a \underset{\cdot}{\backslash} c \Leftrightarrow a \preceq c \underset{\cdot}{/} b \qquad a \dot{\otimes} b \preceq^d c \Leftrightarrow b \preceq^d a \dot{\backslash} c \Leftrightarrow a \preceq^d c \dot{/} b \tag{5}$$

can be expressed in terms of the multiplications as:

$$\begin{aligned} a \underset{\cdot}{\backslash} c &= a^{-1} \dot{\otimes} c = (a \underset{\cdot}{\otimes} c^{-1})^{-1} & \qquad c \underset{\cdot}{/} a &= c \dot{\otimes} a^{-1} = (c^{-1} \underset{\cdot}{\otimes} a)^{-1} \\ a \dot{\backslash} c &= a^{-1} \underset{\cdot}{\otimes} c = (a \dot{\otimes} c^{-1})^{-1} & \qquad c \dot{/} a &= c \underset{\cdot}{\otimes} a^{-1} = (c^{-1} \dot{\otimes} a)^{-1} \end{aligned} \tag{6}$$

3.2 Semimodules over Reflexive Idempotent Semifields

Basic definitions. A semimodule over a semiring is defined in a similar way to a module over a ring [1,8,7] a *left $\mathcal{S}$-semimodule*, $\mathcal{Y}$, is an additive commutative monoid $\langle Y, \oplus, \epsilon_{\mathcal{Y}}\rangle$ endowed with a map $(\lambda, y) \mapsto \lambda \otimes y$ such that $\forall \lambda, \mu \in S,\ \ y, z \in Y$, and following the convention of dropping the symbol for the scalar action and multiplication for the semiring we have:

$$\begin{aligned} (\lambda\mu)y &= \lambda(\mu y) & \qquad \epsilon_S y &= \epsilon_{\mathcal{Y}} \\ \lambda(y \oplus z) &= \lambda y \oplus \lambda z & \qquad e_S y &= x \end{aligned} \tag{7}$$

The definition of a *right $\mathcal{S}$-semimodule*, $\mathcal{X}$, follows the same pattern with the help of a *right action*, $(\lambda, x) \mapsto x \otimes \lambda$ and similar axioms to those of (7.) A $(\mathcal{K}, \mathcal{S})$*-semimodule* is a set M endowed with left $\mathcal{K}$-semimodule and a right $\mathcal{S}$-semimodule structures, and a $(\mathcal{K}, \mathcal{S})$*-bisemimodule* a $(\mathcal{K}, \mathcal{S})$-semimodule such that the left and right multiplications commute. For a left $\mathcal{S}$-semimodule, $\mathcal{Y}$, the left and right multiplications are defined as: $\mathrm{L}_\lambda^{\mathcal{S}} : Y \to Y, y \mapsto \mathrm{L}_\lambda^{\mathcal{S}}(y) = \lambda y$, and $\mathrm{R}_y^{\mathcal{Y}} : S \to Y, \lambda \mapsto \mathrm{R}_y^{\mathcal{Y}}(\lambda) = \lambda y$. And similarly, for a right S-semimodule.

For instance, the semimodule of finite matrices $\mathcal{M}_{g\times m}(\mathcal{S}) = \langle S^{g\times m}, \oplus, \mathcal{E}\rangle$ is a $(\mathcal{M}_g(\mathcal{S}), \mathcal{M}_m(\mathcal{S}))$-bisemimodule for finite g and m, with matrix multiplication-like left and right actions and componentwise addition, and so are the set of column vectors $\mathcal{M}_{m\times 1}(\mathcal{S})$ and the set of row vectors $\mathcal{M}_{1\times g}(\mathcal{S})$. For the completed semifields of $\overline{\mathbb{R}}_{\max,+}$ and $\overline{\mathbb{R}}_{\min,+}$, we have:

$$(A \underset{\cdot}{\otimes} B)_{ij} := \max_{k=1}^{n}(A_{ik} \underset{\cdot}{+} B_{kj}) \qquad (C \overset{\cdot}{\otimes} D)_{ij} := \min_{k=1}^{n}(C_{ik} \overset{\cdot}{+} D_{kj})$$

A left, right $\mathcal{D}$-semimodule $\mathcal{X}$ over an idempotent semiring $\mathcal{D}$ inherits the idempotent law: $\forall v \in X,\ v \oplus v = v$, which induces a *natural order* on the semimodule: $\forall v, w \in X,\ v \leq w \iff v \oplus w = w$, whereby it becomes a $\vee$-semilattice, with $\epsilon_{\mathcal{X}}$ the minimum. In the following we systematically equate idempotent $\mathcal{D}$-semimodules and semimodules over an idempotent semiring $\mathcal{D}$. When $\mathcal{D}$ is a complete idempotent semiring, a left $\mathcal{D}$-semimodule, $\mathcal{X}$ is *complete (in its natural order)* if it is complete as a naturally ordered set and its left and right multiplications are (lower semi)continuous. Trivially, it is also a complete lattice, with join and meet operations given by: $v \leq w \iff v \vee w = w \iff v \wedge w = v$. This extends naturally to right- and bisemimodules.

As in the semiring case, because of the natural order structure, the actions of idempotent semimodules admit residuation: given a complete, idempotent left $\mathcal{D}$-semimodule, $\mathcal{Y}$, we define for all $y, z \in Y$, $\lambda \in D$ the residuals $\left(\mathrm{L}_\lambda^{\mathcal{D}}\right)^{\#} : Y \to Y, z \mapsto \left(\mathrm{L}_\lambda^{\mathcal{D}}\right)^{\#}(z) = \lambda\backslash z$, $\left(\mathrm{R}_y^{\mathcal{Y}}\right)^{\#} : Y \to D, z \mapsto \left(\mathrm{R}_y^{\mathcal{Y}}\right)^{\#}(z) = z/y$, and likewise for a right semimodule, $\mathcal{X}$.

If $\mathcal{D}$ is idempotent (resp. complete), then finite matrix semimodules are idempotent (resp. complete) with the componentwise partial order their natural order. For $\mathcal{D}$ a completed idempotent semifield as per construction 4, the left and right residuals of $\underset{\cdot}{\otimes}$ and $\overset{\cdot}{\otimes}$ are:

$$\begin{aligned}
(A \underset{\cdot}{\backslash} B)_{ij} &= \bigoplus_{k=1}^{m}\left(A_{ki}^{-1} \overset{\cdot}{\otimes} B_{kj}\right) \qquad & (B \underset{\cdot}{/} C)_{ij} &= \bigoplus_{k=1}^{p}\left(B_{ik} \overset{\cdot}{\otimes} C_{jk}^{-1}\right) \qquad (8)\\
(A \overset{\cdot}{\backslash} B)_{ij} &= \bigoplus_{k=1}^{m}\left(A_{ki}^{-1} \underset{\cdot}{\otimes} B_{kj}\right) \qquad & (B \overset{\cdot}{/} C)_{ij} &= \bigoplus_{k=1}^{p}\left(B_{ik} \underset{\cdot}{\otimes} C_{jk}^{-1}\right)
\end{aligned}$$

with summations those of the dioid corresponding to the multiplication.

There is a remarkable operation that changes the character of a semimodule while at the same time reversing its order by means of residuation: let $\mathcal{D}$ be a complete dioid, and $\mathcal{X}$ be a complete right $\mathcal{D}$-semimodule, its *opposite semimodule* is the complete *left* $\mathcal{D}$-semimodule $\mathcal{X}^{\mathrm{op}} = \langle X, \overset{\mathrm{op}}{\oplus}, \bot_{\mathcal{X}^{\mathrm{op}}}\rangle$ with the same underlying set X, addition defined by $(x, z) \mapsto x \overset{\mathrm{op}}{\oplus} z = x \wedge z$ where the infimum is for the natural order of $\mathcal{X}$, bottom element $\bot_{\mathcal{X}^{\mathrm{op}}} = \top_{\mathcal{X}}$, and *left* action: $D \times X \to X \quad (\lambda, x) \mapsto \lambda \overset{\mathrm{op}}{\to} x = x/\lambda$. Consequently, the order of the opposite is the *dual* of the original order.

It is easy to see that $\overline{\mathbb{R}}_{\min,+}$ is precisely the complete, idempotent semiring opposite to $\overline{\mathbb{R}}_{\max,+}$, taken as a semimodule, $\overline{\mathbb{R}}_{\min,+} = (\overline{\mathbb{R}}_{\max,+})^{\mathrm{op}}$ and viceversa, $\overline{\mathbb{R}}_{\max,+} = (\overline{\mathbb{R}}_{\min,+})^{\mathrm{op}}$, since opposition of semimodules is an involution.

Finally, for an element of a semimodule over and idempotent semifield, $x \in X$, we define the *inverse* as $(x^{-1})_i = x_i^{-1}$ (which is not felicitous, given that $x^{-1} \otimes x$

is not defined in general.) However, with the precautions taken for $\overline{\mathbb{R}}_{\max,+}, \overline{\mathbb{R}}_{\min,+}$ we can write: $(x^{-1})_i := -x_i$.

Basic construction of Galois connections over reflexive semimodules. The following construction is due to Cohen et al. [1]. Let $\mathcal{D}$ be a complete dioid; for a bracket $\langle \cdot \mid \cdot \rangle : Y \times X \to Z$ between left and right $\mathcal{D}$-semimodules, $\mathcal{Y}$ and $\mathcal{X}$ respectively, onto a $\mathcal{D}$-bisemimodule $\mathcal{Z}$ and an arbitrary element $\varphi \in Z$, which we call the *pivot*, define the maps, $\cdot^*_\varphi : Y \to X$ and ${}^*_\varphi\cdot : X \to Y$:

$$y^*_\varphi = \bigvee \{\, x \in X \mid \langle y \mid x \rangle \leq \varphi \,\} \qquad {}^*_\varphi x = \bigvee \{\, y \in Y \mid \langle y \mid x \rangle \leq \varphi \,\} \tag{9}$$

Proposition 1 ([1], th. 42). $\left(\cdot^*_\varphi,\ {}^*_\varphi\cdot\right) : \mathcal{Y} \rightharpoonup \mathcal{X}$ *is a* TYPE OI *Galois connection.*

Note that $\mathcal{X}$ and $\mathcal{Y}$ are both already complete lattices as well as free vector spaces. However, the closure lattices $\overline{Y} = {}^*_\varphi(X)$ and $\overline{X} = (\mathcal{Y})^*_\varphi$ do not generally agree with their ambient vector spaces in their joins, but only in their meets. A *reflexive dioid*, $(\mathcal{D}, \varphi)$, is a complete dioid such that $(\langle \cdot \mid \cdot \rangle : D \times D \to D, \varphi)$ with $\langle \lambda \mid \mu \rangle = \lambda\mu$ induces a *perfect Galois connection* under construction (9), that is, a pair of mutually inverse order isomorphisms: $\forall \lambda \in D$, ${}^*_\varphi(\lambda^*_\varphi) = \lambda$, and $({}^*_\varphi\lambda)^*_\varphi = \lambda$. In reflexive dioids $\overline{X}$ and $\overline{Y}$ are actually (join-)subsemimodules of the corresponding spaces ([1], prop. 28).

This construction is affected crucially by the choice of a suitable pivot φ: if we consider the bracket to reflect a *degree of relatedness* between the elements of each pair, only those pairs $(y, x) \in Y \times X$ are considered by the connection whose degree amounts *at most* to φ . Therefore we can think of the pivot as a *maximum degree of existence* allowed for the pairs.

Finally, φ need not be unique: if $(\mathcal{D}, \varphi)$ is reflexive, for any $\lambda \in D$ invertible, $(\mathcal{D}, \varphi\lambda)$ is reflexive. Cohen et al. [1] prove that idempotent semifields are reflexive, and suggest that for the Boolean semiring we must choose $\varphi = 0_{\mathcal{B}}$, the bottom in the order. For other semifields any invertible element may be chosen, e.g. $\varphi = e_{\mathcal{D}}$.

4 Galois Connections Generated by Matrices over Completed Idempotent Semifileds

In this section we provide an easy way to build all possible Galois connections between two semimodules over an idempotent semifield. We use the Moreau notation troughout to prove that it simplifies things considerably. For all of this section, consider a completed, reflexive idempotent semiring $(\mathcal{D}, e_{\mathcal{D}})$, and let $\mathcal{Y}$ and $\mathcal{X}$ be left and right semimodules over $\mathcal{D}$ or its opposite.

Definition 6. *For* $\mathcal{Y} \cong \mathcal{D}^{1\times n}, \mathcal{X} \cong \mathcal{D}^{n\times 1}$ *and bracket* $\langle \cdot \mid \cdot \rangle_{\mathrm{OI}} : Y \times X \to D$, $\langle y \mid x \rangle_{\mathrm{OI}} = y \mathbin{\dot\otimes} x$ *we define a* conjugation *to be the Galois connection of type* OI *obtained from the maps in equation 9, and we write simply:* $(\cdot^*, {}^*\cdot) : \mathcal{Y} \rightharpoonup \mathcal{X}$.

Table 3. Brackets between left and right free semimodules defined over a complete idempotent semifield and its opposite.

With range in $\mathcal{D}$	With range in $\mathcal{D}^{op}$
$\langle \cdot \mid \cdot \rangle_{\mathrm{OI}} : \mathcal{D}^{1\times n} \times \mathcal{D}^{n\times 1} \to \mathcal{D}$ $\langle y \mid x \rangle_{\mathrm{OI}} = y \underset{\cdot}{\otimes} x$	$[\cdot \mid \cdot]_{\mathrm{OI}} : (\mathcal{D}^{op})^{1\times n} \times (\mathcal{D}^{op})^{n\times 1} \to \mathcal{D}^{op}$ $[y \mid x]_{\mathrm{OI}} = y \overset{\cdot}{\otimes} x$
$\langle \cdot \mid \cdot \rangle_{\mathrm{OO}} : \mathcal{D}^{1\times n} \times (\mathcal{D}^{op})^{1\times n} \to \mathcal{D}$ $\langle y \mid x \rangle_{\mathrm{OO}} = y \overset{\cdot}{/} x = y \underset{\cdot}{\otimes} x^*$	$[\cdot \mid \cdot]_{\mathrm{OO}} : (\mathcal{D}^{op})^{1\times n} \times \mathcal{D}^{1\times n} \to \mathcal{D}^{op}$ $[y \mid x]_{\mathrm{OO}} = y \underset{\cdot}{/} x = y \overset{\cdot}{\otimes} x^*$
$\langle \cdot \mid \cdot \rangle_{\mathrm{IO}} : (\mathcal{D}^{op})^{n\times 1} \times (\mathcal{D}^{op})^{1\times n} \to \mathcal{D}$ $\langle y \mid x \rangle_{\mathrm{IO}} = (x \overset{\cdot}{\otimes} y)^* = y^* \underset{\cdot}{\otimes} x^*$	$[\cdot \mid \cdot]_{\mathrm{IO}} : \mathcal{D}^{n\times 1} \times \mathcal{D}^{1\times n} \to \mathcal{D}^{op}$ $[y \mid x]_{\mathrm{IO}} = (x \underset{\cdot}{\otimes} y)^* = y^* \overset{\cdot}{\otimes} x^*$
$\langle \cdot \mid \cdot \rangle_{\mathrm{II}} : (\mathcal{D}^{op})^{n\times 1} \times \mathcal{D}^{n\times 1} \to \mathcal{D}$ $\langle y \mid x \rangle_{\mathrm{II}} = y \overset{\cdot}{\backslash} x = y^* \underset{\cdot}{\otimes} x$	$[\cdot \mid \cdot]_{\mathrm{II}} : \mathcal{D}^{n\times 1} \times (\mathcal{D}^{op})^{n\times 1} \to \mathcal{D}^{op}$ $[y \mid x]_{\mathrm{II}} = y \underset{\cdot}{\backslash} x = y^* \overset{\cdot}{\otimes} x$

By Equation (9): $y^* = y \underset{\cdot}{\backslash} e_{\mathcal{D}}$, ${}^*x = e_{\mathcal{D}} \underset{\cdot}{/} x$. For any other invertible element φ we have the *φ-conjugations*: $y^*_\varphi = y \underset{\cdot}{\backslash} \varphi = y \underset{\cdot}{\backslash} (e_{\mathcal{D}} \overset{\cdot}{\otimes} \varphi) = y^* \overset{\cdot}{\otimes} \varphi$ and ${}^*_\varphi x = \varphi \overset{\cdot}{\otimes} {}^*x$. Hence, the conjugations in $\overline{\mathbb{R}}_{\max,+}$ are: $y^* := -y^{\mathrm{t}}, {}^*x := -x^{\mathrm{t}}$.

Consider Table 3.We claim:

Proposition 2. *1. The brackets in the left column generate all possible types of Galois connections between $\mathcal{Y}$ and $\mathcal{X}$ by composition with adequate conjugations.*

2. The brackets in the right column generate all possible connections between the conjugates of $\mathcal{Y}$ and $\mathcal{X}$ by composition with adequate conjugations.

Proof. For 1) Bracket $\langle \cdot \mid \cdot \rangle_{\mathrm{OI}}$ generates the conjugations above, which are Galois connections of TYPE OI by Proposition 1. $\langle \cdot \mid \cdot \rangle_{\mathrm{II}}$ generates another type OI between $(\mathcal{D}^{op})^{n\times 1}$ and $\mathcal{D}^{n\times 1}$, hence pre-composing with a conjugation between $\mathcal{D}^{1\times n}$ and $(\mathcal{D}^{op})^{n\times 1}$ as defined previously obtains a right adjunction, TYPE II, between $\mathcal{D}^{1\times n}$ and $\mathcal{D}^{n\times 1}$. The procedure is exactly the same for type OO and type IO connections. For 2) the procedure is exactly the same starting from $[\cdot \mid \cdot]_{\mathrm{OI}}$ which is the one generating the Galois connection proper between $(\mathcal{D}^{op})^{1\times n}$ and $(\mathcal{D}^{op})^{n\times 1}$. □

The following proposition states, essentially, that the Galois connections over $\mathcal{D}$ and its opposite are essentially inverses (as expected from the inversion of orders between the opposite semifields):

Proposition 3. *For all brackets above, for $k \in \{$OI,OO,IO,II$\}$ we have:*

$$\langle y \mid x \rangle_k = \left(\left[y^{-1} \mid x^{-1}\right]_k\right)^{-1} \qquad [y \mid x]_k = \left(\langle y^{-1} \mid x^{-1} \rangle_k\right)^{-1} \tag{10}$$

Note that such Galois connections can be built being $\mathcal{D}$ either a scalar or a matrix semiring. Hence, considering the brackets in Table 4, we claim, with a similar proof:

Table 4. Brackets between left and right free semimodules defined over a complete idempotent semifield and its opposite with the aid of matrices defined over each semifield.

With range in $\mathcal{D}$	With range in $\mathcal{D}^{op}$
$\langle \cdot \mid \cdot \rangle^R_{\mathrm{OI}} : \mathcal{D}^{1\times g} \times \mathcal{D}^{m\times 1} \to \mathcal{D}$ $\langle y \mid x \rangle^R_{\mathrm{OI}} = y \underset{\cdot}{\otimes} R \underset{\cdot}{\otimes} x$	$[\cdot \mid \cdot]^R_{\mathrm{OI}} : (\mathcal{D}^{op})^{1\times g} \times (\mathcal{D}^{op})^{m\times 1} \to \mathcal{D}^{op}$ $[y \mid x]^R_{\mathrm{OI}} = y \overset{\cdot}{\otimes} R \overset{\cdot}{\otimes} x$
$\langle \cdot \mid \cdot \rangle^R_{\mathrm{OO}} : \mathcal{D}^{1\times g} \times (\mathcal{D}^{op})^{1\times m} \to \mathcal{D}$ $\langle y \mid x \rangle^R_{\mathrm{OO}} = (y \underset{\cdot}{\otimes} R) \overset{\cdot}{/} x = y \underset{\cdot}{\otimes} R \underset{\cdot}{\otimes} x^*$	$[\cdot \mid \cdot]^R_{\mathrm{OO}} : (\mathcal{D}^{op})^{1\times g} \times \mathcal{D}^{1\times m} \to \mathcal{D}^{op}$ $[y \mid x]^R_{\mathrm{OO}} = (y \overset{\cdot}{\otimes} R) \underset{\cdot}{/} x = y \overset{\cdot}{\otimes} R \overset{\cdot}{\otimes} x^*$
$\langle \cdot \mid \cdot \rangle^R_{\mathrm{IO}} : (\mathcal{D}^{op})^{g\times 1} \times (\mathcal{D}^{op})^{1\times m} \to \mathcal{D}$ $\langle y \mid x \rangle^R_{\mathrm{IO}} = (x \overset{\cdot}{\otimes} {}^*R \overset{\cdot}{\otimes} y)^* = y^* \underset{\cdot}{\otimes} R \underset{\cdot}{\otimes} x^*$	$[\cdot \mid \cdot]^R_{\mathrm{IO}} : \mathcal{D}^{g\times 1} \times \mathcal{D}^{1\times m} \to \mathcal{D}^{op}$ $[y \mid x]^R_{\mathrm{IO}} = (x \underset{\cdot}{\otimes} {}^*R \underset{\cdot}{\otimes} y)^* = y^* \overset{\cdot}{\otimes} R \overset{\cdot}{\otimes} x^*$
$\langle \cdot \mid \cdot \rangle^R_{\mathrm{II}} : (\mathcal{D}^{op})^{g\times 1} \times \mathcal{D}^{m\times 1} \to \mathcal{D}$ $\langle y \mid x \rangle^R_{\mathrm{II}} = y \overset{\cdot}{\backslash} (R \underset{\cdot}{\otimes} x) = y^* \underset{\cdot}{\otimes} R \underset{\cdot}{\otimes} x$	$[\cdot \mid \cdot]^R_{\mathrm{II}} : \mathcal{D}^{g\times 1} \times (\mathcal{D}^{op})^{m\times 1} \to \mathcal{D}^{op}$ $[y \mid x]^R_{\mathrm{II}} = y \underset{\cdot}{\backslash} (R \overset{\cdot}{\otimes} x) = y^* \overset{\cdot}{\otimes} R \overset{\cdot}{\otimes} x$

Proposition 4. *1. For a given $R \in \mathcal{M}_{g\times m}(\mathcal{D})$, the brackets in the left column generate all possible types of Galois connections between the appropriate $\mathcal{Y}$ and $\mathcal{X}$ by composition with adequate conjugations.*

2. For a given $R \in \mathcal{M}_{g\times m}(\mathcal{D}^{op})$, the brackets in the right column generate all possible connections between the appropriate $\mathcal{Y}$ and $\mathcal{X}$ by composition with adequate conjugations.

Proof. The proof is straightforward following the steps of proposition 2 and a property similar to that for the brackets above. The proof for TYPE OI Galois connections can be found in [1], §4.5, as well as that of TYPE IO. □

5 Conclusion

In this paper we have provided algebraic formulae for the construction of Galois connections of all four different types viz. left and right adjunctions, Galois connections proper and co-Galois connections, between semimodules over idempotent, reflexive semifields. Although such semifields turn out to be incomplete, we have supplied a construction allowing their completion and, further, a notation, reminiscent of one introduced by Moreau, for expressing all Galois connection operators in matrix algebra.

The main scheme of combining a basic Galois connection proper plus the Cohen-Gaubert-Quadrat conjugation is already looming in [1,5]. Similarly, the use of the Moreau notation is already present in [10] in relation to co-Galois connections (TYPE IO) and the completions of certain idempotent semigroups but was not explored systematicatically there. Of course, this is exactly the way the right-axiality ($R_{\forall}{}^{\exists}$) and the co-Galois connection ($R_{-}{}^{-}$) where introduced in [4], but only for subsets of 2^G and 2^M, whose generalisation for other semirings is not straightforward. All in all, this shows directly that $\mathcal{K}$-Formal Concept Analysis is just one of the cases here described and indirectly the same holds for standard Formal Concept Analysis.

Acknowledgements We would like to thank M. Erné for providing initial input for Galois connection diagrams in section 2.

References

1. G. Cohen, S. Gaubert, and J.-P. Quadrat. Duality and separation theorems in idempotent semimodules. *Linear Algebra and Its Applications*, 379:395–422, 2004.
2. B. Davey and H. Priestley. *Introduction to lattices and order.* Cambridge University Press, Cambridge, UK, 2nd edition, 2002.
3. M. Erné. *Adjunctions and Galois connections: Origins, History and Development*, volume 565 of *Mathematics and Its Applications*, pages 1–138. Kluwer Academic, Dordrecht, Boston and London, 2004.
4. M. Erné, J. Koslowski, A. Melton, and G. Strecker. A primer on Galois connections. In A. Todd, editor, *Proceedings of the 1991 Summer Conference on General Topology and Applications in Honor of Mary Ellen Rudin and Her Work*, volume 704 of *Annals of the New York Academy of Sciences*, pages 103–125, Madison, WI, USA, June 1991 1993. New York Academy of Science.
5. S. Gaubert. Two lectures on max-plus algebra. Support de cours de la 26–iéme École de Printemps d'Informatique Théorique, May 1998. http://amadeus.inria.fr/gaubert/papers.html.
6. G. Gediga and I. Dütsch. Approximation operators in qualitative data analysis. Technical Report CS-03-01, Department of Computer Science, Brock University, St. Catharines, Ontario, Canada, May 2003.
7. J. S. Golan. *Power Algebras over Semirings. With Applications in Mathematics and Computer Science*, volume 488 of *Mathematics and its applications.* Kluwer Academic, Dordrecht, Boston, London, 1999.
8. J. S. Golan. *Semirings and Their Applications.* Kluwer Academic, 1999.
9. J. J. Moreau. Inf-convolution, sous-additivité, convexité des fonctions numériques. *J. Math. pures et appl.*, 49:109–154, 1970.
10. I. Singer. *Abstract Convex Analysis.* Monographs and Advanced Texts. Wiley-Interscience, 1997.
11. I. Singer. (*,s)-dualities. *Journal of Mathematical Sciences*, 115(4):2506–2541, 2003.
12. I. Singer. Some relations between linear mappings and conjugations in idempotent analysis. *Journal of Mathematical Sciences*, 115(5):2610–2630, 2003.
13. P. Taylor. *Practical Foundations of Mathematics.* Number 59 in Cambridge studies in advanced mathematics. Cambridge University Press, 1999.
14. F. J. Valverde-Albacete and C. Peláez-Moreno. Towards a generalisation of Formal Concept Analysis for data mining purposes. In R. Missaoui and J. Schmid, editors, *Concept Lattices. Proceedings of the International Conference on Formal Concept Analysis, ICFCA06, (Dresden, Germany)*, volume 3874 of *LNAI*, pages 161–176, Berlin, Heidelberg, 2006. Springer.

Dynamical modification of context for an iterative and interactive information retrieval process on the web.

Emmanuel Nauer and Yannick Toussaint

LORIA – UMR 7503. B.P. 239, F-54506 Vandœuvre-lès-Nancy cedex, France.
{Emmanuel.Nauer,Yannick.Toussaint}@loria.fr

Abstract. This paper presents an iterative and interactive information retrieval system to search on the web using formal concept analysis (FCA). FCA provides a natural way to organize objects according to their properties and it has been used in recent works to organise in a more convenient manner answers provided by a search engine. The navigation into the lattice helps the user explore a structured and synthetic result. Such a lattice contains concepts that are relevant and some others that are not relevant for a given information retrieval task. We introduce lattices into an interactive and iterative system. The user expresses his negative or positive agreement with some concept of the lattice, in respect with his objective of information retrieval. These user choices are converted into operations over the lattice so to make the context change and to better fit user needs.
Keywords: information retrieval, concept lattices, user feedback.

1 Motivation

Formal concept analysis (FCA) provides a natural way to organise objects according to their properties. In the framework of Information Retrieval (IR), FCA has been applied to query refinement, ranking, documents classification, etc. In a web context, some recent works [5, 1] proposed to built a concept lattice starting from title and snippet words of documents returned by a search engine (like GOOGLE), in order to organise them in a structured and synthetic result. To the best of our knowledge, no work has been done on the evolution of the hierarchy during an IR process. The hierarchy is built once and gives an abstract view of the set of documents, in which the user may navigate. However, there is no possibility to change the lattice which may contain numerous non relevant concepts (and documents). In the same time, relevant documents may exist outside the initial set of documents used to build the lattice. So, making the initial set of documents –and the lattice– evolve is a way to tune the results of the IR system to the user needs.

In the domain of IR, many works have be done to embed the user inside the IR process in order to improve the performances of the systems. In these works, user feedback directly impacts on the IR system. This paper presents the CreChainDo system which plugs the user feedback approach on a lattice navigation. The user explores the lattice, and identifies relevant or irrelevant concepts. The user feedback is converted into a reduction or an extension of the context of the lattice and a new lattice is built.

The paper is organised as follows: section 2 shows how FCA and user feedback improve IR. Section 3 details the CreChainDo system and section 4 gives a concrete example of CreChainDo at work. Future directions for this work conclude the paper.

2 Improving IR using FCA

2.1 Formal concept analysis

	`Web`	`IR`	`SearchEngine`	`WebDirectory`	`Yahoo!`	`Altavista`	`Google`
$\mathtt{d}_1$	×	×	×	×	×		
$\mathtt{d}_2$	×	×	×				
$\mathtt{d}_3$	×	×	×				
$\mathtt{d}_4$	×			×			
$\mathtt{d}_5$		×	×			×	×
$\mathtt{d}_6$		×	×				

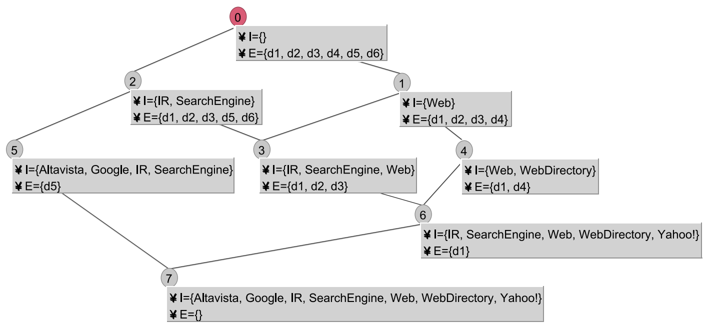
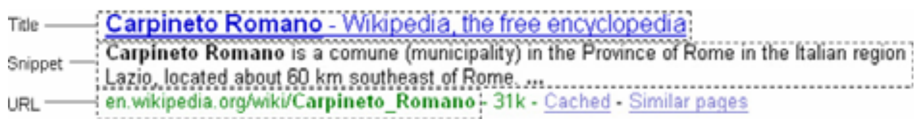

0
I={}
E={d1, d2, d3, d4, d5, d6}
2
I={IR, SearchEngine}
E={d1, d2, d3, d5, d6}
1
I={Web}
E={d1, d2, d3, d4}
5
I={Altavista, Google, IR, SearchEngine}
E={d5}
3
I={IR, SearchEngine, Web}
E={d1, d2, d3}
4
I={Web, WebDirectory}
E={d1, d4}
6
I={IR, SearchEngine, Web, WebDirectory, Yahoo!}
E={d1}
7
I={Altavista, Google, IR, SearchEngine, Web, WebDirectory, Yahoo!}
E={}

Fig. 1. A binary context and the corresponding concept lattice.

FCA is a mathematical approach to data analysis based on lattice theory. A *formal context* is a triple $\mathcal{K} = (G, M, I)$, where G is a set of individuals (called *objects*), M a set of properties (called *attributes*) and I the relation on $G \times M$ stating that an object is described by a property [11]. Table in the left-hand side of Fig. 1 gives an example of context: G is a set of 6 documents ($\mathtt{d}_1$,...,$\mathtt{d}_6$) and M the set of properties composed of 7 keywords describing the documents. A *formal concept* is a pair (I, E), where E is a maximal set of individuals (called *extent*) and I is a maximal set of properties (called *intent*) shared by this extent. For instance, $(\{\mathtt{IR}, \mathtt{SearchEngine}, \mathtt{Web}\}, \{\mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3\})$ is a concept (see diagram in the right-hand side of Fig. 1).

Furthermore, the set $\mathcal{C}_\mathcal{K}$ of all formal concepts of the context $\mathcal{K} = (G, M, I)$ is partially ordered by extent inclusion also called the *specialization* (denoted $\leq_\mathcal{K}$) between concepts. $\mathcal{L} = \langle \mathcal{C}_\mathcal{K}, \leq_\mathcal{K} \rangle$ is a complete lattice, called the *concept lattice.* The lattice $\mathcal{L}$ can be drawn as a Hasse diagram where nodes are concepts, and line segments specialization links. Fig. 1 illustrates a context and its corresponding lattice. The top concept contains all the documents ; its intent is empty because there is no common property shared by all the documents. On the opposite, the bottom concept is defined by the set of all properties and its extent is empty as none of the documents is described by all the properties. A great number of algorithms has been proposed for the construction of concept lattices, see [11]. For our application, we use coron, a software platform implementing a rich



Carpineto Romano - Wikipedia, the free encyclopedia
Carpineto Romano is a comune (municipality) in the Province of Rome in the Italian region Lazio, located about 60 km southeast of Rome. ...
en.wikipedia.org/wiki/**Carpineto_Romano** - 31k - Cached - Similar pages

Fig. 2. A GOOGLE document returned for the query *"carpineto romano"*.

set of algorithmic methods for symbolic datamining, including concept lattice construction algorithms [19].

2.2 Formal concept analysis for information retrieval

FCA has been applied to solve various core problems of IR systems (a fine state of the art is presented in [5]). Lattice based IR follows the idea that a concept intent can be seen as a query and its extent as the retrieved documents. The underlying idea is that the neighbour concepts of a query concept can be seen as the minimal changes for a query reformulation. The REFINER system [3] exploits these properties, by building only a part of the lattice around the query concept and by displaying it to the user.

Concept lattices are also good candidates for hybrid IR systems based on querying and navigation [6, 8]. The user query is classified into the lattice. This concept is the starting point for a navigation into the hierarchical structure. In this framework, [2] proposes to enhance the lattice structure by exploiting a subsumption hierarchy about the set of terms describing the documents, in the form of a thesaurus. Two specific functionalities for querying and navigating into the lattice are proposed in [9]. The first one, a query-by-example function retrieves documents which own common properties shared by a set of documents. The second one, a similarity measure between concepts (based on their intensions and extensions) ranks similar concepts to a given one.

Finally, lattices are also used for ordering documents returned by an IR system. Approaches have been proposed for document ranking [4], or even for documents re-organisation. The CREDO (standing for Conceptual REorganisation of DOcuments) system [5] (or its adaptation CREDINO for PDA [1]) uses a lattice for clustering and navigating into a set of documents returned by GOOGLE on a given query. Each GOOGLE document, as the one given on Fig. 2, is composed of a title, a snippet and an URL. CREDO builds two binary contexts. The first level of the hierarchy results from the *document* $\times$ *title words* context, whereas the next levels result from the *document* $\times$ *(title + snippet words)* context. In this hierarchy, the intent of a concept is a set of words and its related extent is the set of documents containing all the words of the intent. The initial flat list of documents given by GOOGLE is now organised in a strutured way. The navigation into this hierarchy helps the user to explore the set of documents. We improve the CREDO approach adding interactive capabilities.

2.3 Iterative and interactive information retrieval process

The goal of an IR system is to provide a user with information about a given need. In most IR systems, and especially web search engines, the IR process consists in the submission of a query representing the user need. As a query is a reduce representation of the user's need, some IR systems exploit additional information. Relevance Feedback (RF) [15] provides more information on the search and is known to be effective for improving retrieval accuracy [16].

There are two kinds of RF: *explicit RF* and *implicit RF*. In explicit RF, the user has to give explicitly his feedback to the IR system by, for example, entering keywords, answering to specific questions, marking a subset of retrieved documents as relevant or irrelevant [12], annotating documents [7], etc. In implicit RF, IR systems do not propose any specific interactions for the feedback. The feedback is deduced by the IR system according to all the *implicit* interactions of the user [17]. For example, in a query system, the results returned on a first query may generally not be satisfactory. Often, the user may need to modify his query and to view ranked documents with many iterations before the information need is completely satisfied. All these interactions may be exploited for feedback. Query reformulations may, for example, disambiguate the context of polysemic words [10]. We refer to [13] for a classification of implicit feedback techniques in link with the major works of the domain.

The IR process proposed in CreChainDo implements an explicit RF. The user interacts iteratively with the system for evaluating if a concept is relevant or not. His feedback is linked to a modification of the context used to built the lattice.

3 The CreChainDo system

3.1 Principles

To take into account the iterativity and interactivity of the IR process, CreChainDo [1] integrates a dynamical context modification approach. In the first step of the IR process, the user submits a query and a lattice is built. Then, the user may perform choices over the lattice. Like in IR systems in which the user can accept/reject documents or set of documents, CreChainDo offers the user to select concepts which are relevant and concepts which are not relevant.

Definition 1. *A concept is* relevant *with respect to the user need, if a query Q, resulting from the conjunction of all the words of its intent, makes more precise the user need.*

The underlying idea of this definition is that the lattice do not necessarily contain, at a given time, all the relevant documents that can be found on the web, and that a new search engine interrogation on Q may return new relevant documents. This definition does not take into account the concept extent because the extent of a relevant concept does not necessarily contain only relevant documents.

Definition 2. *A concept is* not relevant *according a user need, if the documents contained in its extent are not relevant for the user need, and if none of the words, defining its intent, can take part of the intent of a relevant concept.*

[1] The experimental protoype is available at http://intoweb.loria.fr/CreChainDo

The CreChainDo interface (see Fig. 4, afterwards) proposes to the user, for each concept of the hierarchy, to click on the icon ➡ if relevant and ✖ if not relevant. Each user action acts on a context modification and a new lattice is computed. In CreChainDo, the IR process is a succession of context: $K_0 \rightarrow K_1 \rightarrow \ldots \rightarrow K_i \rightarrow K_{i+1} \rightarrow \ldots$. We note $K_i = (G_i, M_i, I_i)$ the context related to the step i. $K_0 = (G_0, M_0, I_0)$ denotes the initial empty context: $G_0 = \emptyset$, and thus $M_0 = \emptyset$ and $I_0 = \emptyset$.

3.2 Context extension

We denote by Q a user query. Q is a non empty set of words : $Q = \{word_i | i > 0\}$. We denote by $DOC(Q)$ the set of documents returned by a search engine to the query Q: $DOC(Q) = \{doc_j | j \geq 0\}$.

Definition 3. *The extension of the context K_i to K_{i+1}, noted $K_i \xrightarrow{+Q} K_{i+1}$, is done by adding the result returned by a search engine to the query Q :*

- *new documents are added to the objects set of the context : $G_{i+i} = G_i \cup DOC(Q)$;*
- *new words are added to the properties set of the context : $M_{i+i} = M_i \cup \{word_i | i \in DOC(Q)\}$;*
- *the relation I_i between G_i and M_i is extended to the relation I_{i+1} between G_{i+1} and M_{i+1}*

In a K_{i+1} *extended* context, objects from G_i are described exactly with the same properties. Therefore, for all $(g, m) \in I_i$, if $g \in G_{i+1}$ then $(g, m) \in I_{i+1}$. For all concept $(g, m_i) \in K_i$, there exists a concept $(g, m_{i+1}) \in K_{i+1}$ where $m_i \subseteq m_{i+1}$. The intent remains unchanged while extent may increase. This ensures a progressive lattice evolution between two steps.

There is two cases, where the context is extended:

- the user submits a new query: the user query Q_0 ($K_0 \xrightarrow{Q_0} K_1$) defines a new context but the user may introduce at anytime in the IR process a new query Q_i (adding new words in the search) between two steps ($K_i \xrightarrow{+Q_i} K_{i+1}$).
- the user evaluates a concept C as relevant for his need: a query composed of the conjunction of all the words defining the intent of C is submitted to a search engine, the context is extended by $K_i \xrightarrow{+intent(C)} K_{i+1}$.

3.3 Context reduction

The objective of a context reduction is to remove from the lattice, concepts that the user evaluates as non relevant for his need.

Definition 4. *The reduction of the context K_i to K_{i+1}, noted $K_i \xrightarrow{-C} K_{i+1}$, is done as follows:*

– *the documents which are in* C *extent are removed from the context:* $G_{i+i} = G_i \setminus extent(C)$*;*
– *the words which are in* C *intent are added to a stopwords list (see 3.4). This implies that* $M_{i+i} = M_i \setminus \{word_i | i \in intent(C)\}$*;*
– *the relation* I_{i+1} *between* G_{i+1} *and* M_{i+1} *is obtained by deleting into* I_i *the lines associated to the documents forming* $extent(C)$ *and the columns with the words of* $intent(C)$*.*

3.4 Implementation of CreChainDo

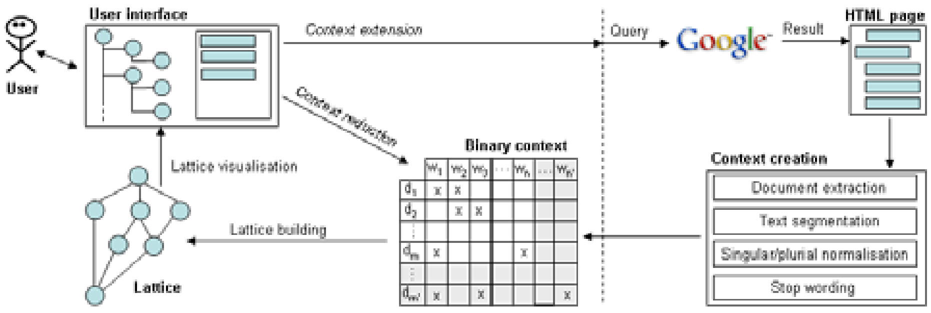
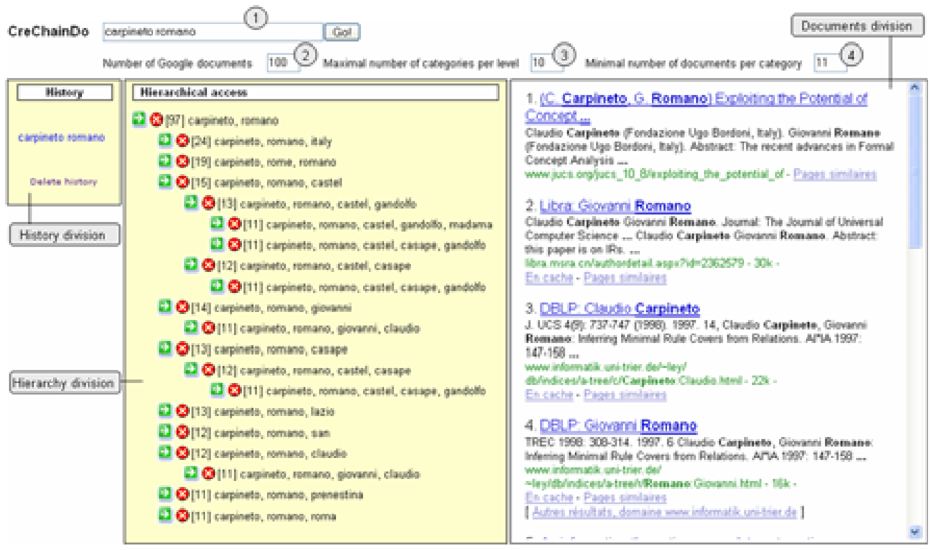


Fig. 3. The CreChainDo architecture.

The architecture of CreChainDo is given on Fig. 3. CreChainDo queries google which returns a web page containing the query result. This page is parsed for extracting a set of documents, as the one given as example on Fig. 2. For each document, the title, the snippet and the URL is extracted. The new documents are added to G_i (the URL is used as key for removing duplicated documents). Creating a context from textual data in CreChainDo follows the classical approach of automatic indexing proposed by [20]. Each document (title ans snippet) is segmented in single words. The singular/plurial inflexional variations are joined using the major rules of variations: a finishing `s` (day → days, etc.), `an` → `en` (man → men, etc.), `y` → `ies` (baby → babies, etc.), `fe` → `ves` (wife → wives, etc.), etc. According to these rules, the vocabulary is normalised by keeping the most frequent form. This avoids the dispersion of the vocabulary. The lattice is directly impacted by reinforcing some concept extensions. Finally, we use the stopword list coming from the Snowball project [18] to eliminate grammatical words (like *"a"*, *"the"*, *"of"*, etc.) and the stopwords list related to negative words identified by the user, to eliminate words that do not represent the user need. All the remaining words are used to create a binary context *document* × *words* (there is no specific word selection). Once the context is built, coron is used to compute the new lattice [19] and a HTML/javascript page is generated and is sent to the web user interface.



carpineto romano

Number of Google documents 100 Maximal number of categories per level 10 Minimal number of documents per category 11

carpineto romano

- [97] carpineto, romano
  - [24] carpineto, romano, italy
  - [19] carpineto, rome, romano
  - [15] carpineto, romano, castel
    - [13] carpineto, romano, castel, gandolfo
      - [11] carpineto, romano, castel, gandolfo, madama
      - [11] carpineto, romano, castel, casape, gandolfo
    - [12] carpineto, romano, castel, casape
      - [11] carpineto, romano, castel, casape, gandolfo
  - [14] carpineto, romano, giovanni
    - [11] carpineto, romano, giovanni, claudio
  - [13] carpineto, romano, casape
    - [12] carpineto, romano, castel, casape
      - [11] carpineto, romano, castel, casape, gandolfo
  - [13] carpineto, romano, lazio
  - [12] carpineto, romano, san
  - [12] carpineto, romano, claudio
    - [11] carpineto, romano, giovanni, claudio
  - [11] carpineto, romano, prenestina
  - [11] carpineto, romano, roma

1. (C. **Carpineto**, G. **Romano**) Exploiting the Potential of Concept ...
Claudio **Carpineto** (Fondazione Ugo Bordoni, Italy). Giovanni **Romano** (Fondazione Ugo Bordoni, Italy). Abstract: The recent advances in Formal Concept Analysis ...
www.jucs.org/jucs_10_8/exploiting_the_potential_of - Pages similaires

2. Libra: Giovanni **Romano**
Claudio **Carpineto** Giovanni **Romano**. Journal: The Journal of Universal Computer Science ... Claudio **Carpineto** Giovanni **Romano**. Abstract: this paper is on IRs. ...
libra.msra.cn/authordetail.aspx?id=2362579 - 30k -
En cache - Pages similaires

3. DBLP: Claudio **Carpineto**
J. UCS 4(9): 737-747 (1998). 1997. 14, Claudio **Carpineto**, Giovanni **Romano**: Inferring Minimal Rule Covers from Relations. AI*IA 1997: 147-158 ...
www.informatik.uni-trier.de/~ley/db/indices/a-tree/c/**Carpineto**:Claudio.html - 22k -
En cache - Pages similaires

4. DBLP: Giovanni **Romano**
TREC 1998: 308-314. 1997. 6 Claudio **Carpineto**, Giovanni **Romano**: Inferring Minimal Rule Covers from Relations. AI*IA 1997: 147-158 ...
www.informatik.uni-trier.de/~ley/db/indices/a-tree/r/**Romano**:Giovanni.html - 16k -
En cache - Pages similaires
[Autres résultats, domaine www.informatik.uni-trier.de]

Fig. 4. The CreChainDo user interface, after a request on "carpineto romano".

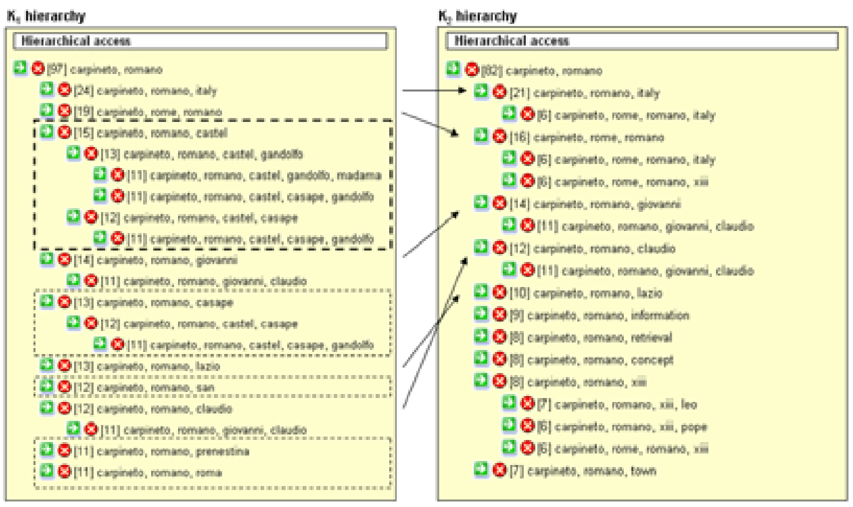
The user interface (see Fig. 4) is divided in 4 parts:

- on the top, the request interface allows the user to (1) submit a query and to specify some parameters like (2) the number of documents wanted to be returned by GOOGLE. Two parameters related to the hierarchy construction and output are available: (3) the maximal number of concepts (`Mnc`) displayed at each level of the hierarchy and (4) the minimal number of documents (`mnd`) contained in the extent of the concepts (this number can be an expressed by a number, or in percentage of the number of documents of the context).
- on the middle of the interface, the hierarchy resulting from a top-down exploration of the lattice, presents the concept intents in a tree view. We choose that all the concepts of the lattice satisfying the `Mnc` and `mnd` parameters will be displayed.
- on the right-hand side of the interface, a division is designed to display the documents related to the concepts. The content of this division is modified dynamically when the user puts his mouse over a concept of the hierarchy. All the documents being in the extent of the selected concept are displayed.
- on the left-hand side, there is an history division, in which, all the users actions are stored: new query, accepting or rejecting a concept. In the future, the objective of this division is that the user could go back on some of his decisions, as propoposed in the EXALEAD (http://www.exalead.fr/) web search engine.

4 Impact of the user interactions over the hierarchical structure.

This section presents and discuss some experiments. Let say that the user searches for documents about the two italian researchers *Carpineto* and *Romano*. This kind of task is very usual for domain analysis or scientific survey. Let *"carpineto romano"* the first query submitted. The resulting hierarchy obtained by extending the initial empty context K_0 by $K_0 \xrightarrow{+\text{"}carpineto\ romano\text{"}} K_1$ is presented on the left-hand side of Fig. 5. We can notice that the top concept contains only 97 documents instead of 100 requested. The reason is that 3 of the 100 documents returned by GOOGLE do not contain the 2 words in the title nor in the snippet.

4.1 Rejecting a non relevant concept



- [97] carpineto, romano
  - [24] carpineto, romano, italy
  - [19] carpineto, rome, romano
  - [15] carpineto, romano, castel
    - [13] carpineto, romano, castel, gandolfo
      - [11] carpineto, romano, castel, gandolfo, madama
      - [11] carpineto, romano, castel, casape, gandolfo
    - [12] carpineto, romano, castel, casape
      - [11] carpineto, romano, castel, casape, gandolfo
  - [14] carpineto, romano, giovanni
    - [11] carpineto, romano, giovanni, claudio
  - [13] carpineto, romano, casape
    - [12] carpineto, romano, castel, casape
      - [11] carpineto, romano, castel, casape, gandolfo
  - [13] carpineto, romano, lazio
  - [12] carpineto, romano, san
  - [12] carpineto, romano, claudio
    - [11] carpineto, romano, giovanni, claudio
  - [11] carpineto, romano, prenestina
  - [11] carpineto, romano, roma

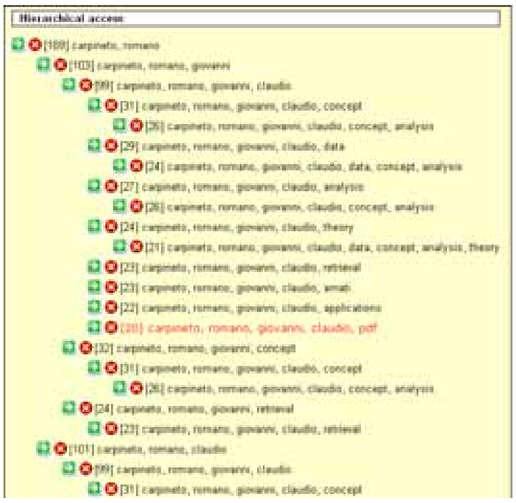
- [82] carpineto, romano
  - [21] carpineto, romano, italy
    - [6] carpineto, rome, romano, italy
  - [16] carpineto, rome, romano
    - [6] carpineto, rome, romano, italy
    - [6] carpineto, rome, romano, xiii
  - [14] carpineto, romano, giovanni
    - [11] carpineto, romano, giovanni, claudio
  - [12] carpineto, romano, claudio
    - [11] carpineto, romano, giovanni, claudio
  - [10] carpineto, romano, lazio
  - [9] carpineto, romano, information
  - [8] carpineto, romano, retrieval
  - [8] carpineto, romano, concept
  - [8] carpineto, romano, xiii
    - [7] carpineto, romano, xiii, leo
    - [6] carpineto, romano, xiii, pope
    - [6] carpineto, rome, romano, xiii
  - [7] carpineto, romano, town

Fig. 5. From K_1 hierarchy to K_2 hierarchy by $K_1 \xrightarrow{-\text{"}carpineto\ romano\ castel\text{"}} K_2$.

The user can quickly see that the hierarchy contains non relevant concepts for a search about *Claudio Carpineto* and *Giovanni Romano*. Let C, with $intent(C) = \{carpineto, romano, castel\}$, be a non relevant concept. Clicking on ⊗ in front of *"carpineto, romano, castel"* in the user interface produces the K_2 hierarchy presented on the right-hand side of Fig. 5. The K_2 hierarchy contains now only 82 documents. In the tree-view display, the effect is a subtree deletion (but in the lattice, these concepts are either removed or merely adapted if they are linked to another part of the lattice). Eliminating documents and terms of a context can affect the whole hierarchy. As illustrated on the left-hand side of Fig. 5, all the framed subtrees have been deleted and some concept extents have been reduced. For example, 3 documents have been deleted in the extent of the *"carpineto, romano, italy"* concept.

4.2 Accepting a relevant concept

Let C, with $intent(C) = \{carpineto, romano, giovanni, claudio\}$ be a relevant concept. Clicking on ➡ in front of C in the user interface produces the K_3 hierarchy presented on Fig. 6. The 87 new documents that have been added produce a more fine grain structuration of the hierarchy.

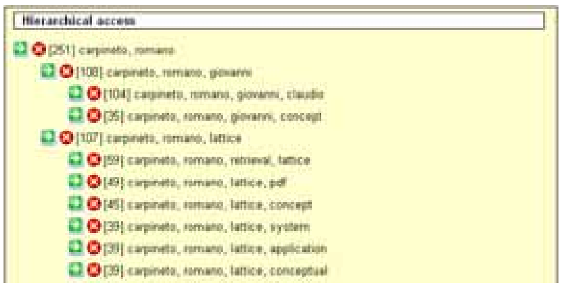


```
Hierarchical access
[189] carpineto, romano
    [103] carpineto, romano, giovanni
        [99] carpineto, romano, giovanni, claudio
            [31] carpineto, romano, giovanni, claudio, concept
                [26] carpineto, romano, giovanni, claudio, concept, analysis
            [29] carpineto, romano, giovanni, claudio, data
                [24] carpineto, romano, giovanni, claudio, data, concept, analysis
            [27] carpineto, romano, giovanni, claudio, analysis
                [26] carpineto, romano, giovanni, claudio, concept, analysis
            [24] carpineto, romano, giovanni, claudio, theory
                [21] carpineto, romano, giovanni, claudio, data, concept, analysis, theory
            [23] carpineto, romano, giovanni, claudio, retrieval
            [23] carpineto, romano, giovanni, claudio, email
            [22] carpineto, romano, giovanni, claudio, applications
            [20] carpineto, romano, giovanni, claudio, pdf
        [32] carpineto, romano, giovanni, concept
            [31] carpineto, romano, giovanni, claudio, concept
                [26] carpineto, romano, giovanni, claudio, concept, analysis
        [24] carpineto, romano, giovanni, retrieval
            [23] carpineto, romano, giovanni, claudio, retrieval
    [101] carpineto, romano, claudio
        [99] carpineto, romano, giovanni, claudio
            [31] carpineto, romano, giovanni, claudio, concept
```

Fig. 6. Part of the K_3 hierarchy, after $K_2 \xrightarrow{+''carpineto\ romano\ giovanni\ claudio''} K_3$.

4.3 New query submission

Sometimes, during the IR process, the user has to reformulate his need. In CreChainDo, each new query submitted to the system will continue to add documents onto the context. For example, the part of the hierarchy presented on Fig. 7 concerns an extension of K_3 by a new query on *"carpineto romano lattice"*: ($K_3 \xrightarrow{+''carpineto\ romano\ lattice''} K_4$).



```
Hierarchical access
[251] carpineto, romano
    [108] carpineto, romano, giovanni
        [104] carpineto, romano, giovanni, claudio
        [35] carpineto, romano, giovanni, concept
    [107] carpineto, romano, lattice
        [59] carpineto, romano, retrieval, lattice
        [49] carpineto, romano, lattice, pdf
        [45] carpineto, romano, lattice, concept
        [39] carpineto, romano, lattice, system
        [39] carpineto, romano, lattice, application
        [39] carpineto, romano, lattice, conceptual
```

Fig. 7. Part of the K_4 hierarchy, after $K_3 \xrightarrow{+''carpineto\ romano\ lattice''} K_4$.

4.4 Discussion

Three possible types of interaction are available in CreChainDo.

Rejecting a non relevant concept is very useful when a lattice contains concepts which do not immediatly focus on information satisfying the user. This type of lattice is obtained for query with polysemic words, but also if the vocabulary of GOOGLE documents is very scattered. To limit this problem, building concepts on a restricted set of properties –as it is done in CREDO for the first level of the hierarchy– is a possible solution. But in this case, non relevant documents or non relevant concepts are even present into the hierarchy. Therefore, there is a real need to *clean* the hierarchy. Eliminating the noise allows indirectly to focus on relevant documents.

Accepting a relevant concept is a way to add new documents, more specific that those obtained until now, into the context. This is a significant extension of CREDO. Indeed, the hierarchy proposed by CREDO is depth-limited and the degree of specialisation of concepts is limited as well. Indeed, the lattice is built only on 100 documents and the vocabulary used in these documents does not necessarily cover all possible various topics related to the words used to query the system. Moreover, the extent of some concepts can be very small, due to the distribution of the 100 documents over all the concept. In CreChainDo, if a concept C contains few documents –as it is the case for concepts directly subsuming the bottom concept– accepting C will extend the sub-hierarchy starting from C. More specific subconcepts of C will be generated by the construction of a new lattice. So, the depth of the hiearchy and the degree of specialisation are less limited.

Submitting a new query offers the user to reformulate his need by himself, without being restricted on the lattice concepts. So, the user can submit multiple queries to the system and all the results returned by GOOGLE will be synthesised into a same hierarchy. This approach seems to be very interesting for merging results coming from multiple query reformulations and can probably also be applied for merging results coming from multiple information sources, as it is done, for example, by meta search engines.

4.5 Future work

First of all, the iterative approach proposed by CreChainDo, has to be evaluated and validated. More generally, we think that many problems, concerning the IR process, have to be studied concretly. For example: how is it possible to take into account the change of the user need during the process? What are the impacts of our choices for the context modifications on the IR process? For example, the context extension on all the words of a relevant concept is possibly too specific: a context extension restricted on specific words of the relevant concept intent (i.e words which are not inherited from more generic concepts) may be an alternative. These questions, about the CreChainDo IR process, are strongly connected to the lattice construction and the dynamical modification of context.

The *quality* of hierarchies, built according to various strategies like those proposed in CREDO and CreChainDo, has to be examined as well because they play a central role in this type of IR process. We also think that some parts of CreChainDo lattices follows a specific organisation and that the study of the lattice properties is a way to improve the hierarchy quality. For example, in Fig. 6, the concept *"[99] carpineto, romano, giovanni, claudio"* is subsumed by the two concepts *"[101] carpineto, romano, giovanni"* and *"[103] carpineto, romano, claudio"*. We agree that a multiple inheritance favors the access to a concept, which is thus reachable by several paths. Nevertheless, the interest of the two concepts *"[101] carpineto, romano, giovanni"* and *"[103] carpineto, romano, claudio"* can be discussed with regards to (1) their connexions with the *"[99] carpineto, romano, giovanni, claudio"* one and of (2) the very high similarity of their extents (containing *[99]*, *[101]* and *[103]* documents). This difference mainly results from the exploitation of the snippets, which boundaries are not wide enough to cover all the 4 words (but these missing words appear in the complete document).

Finally, an attractive direction concerns the potential of synthesis of lattice for a set of search engine results. This approach can be applied for a same query submited to multiple search engines, or also for multiple queries. In the framework of CreChainDo, all the user interactions which are stored in the history could be exploited for an implicit RF [14]. According to the set of positive words (extracted from user queries and relevant concepts) and to the set of negative words (extracted from user non relevant concept), multiple queries can be submitted to GOOGLE. These queries could automatically be generated from all the possible conjunctions of positive words and from a filtering on all the negative words.

5 Conclusion

The CreChainDo system, presented in this paper, is an innovative application of FCA for IR on the web. The user feedback approach has been plugged on the exploration of a lattice used to structure a flat answers list of documents returned by a search engine. With CreChainDo, the user can act directly on the lattice, to make it evolve by a dynamical modification of context. Cleaning the lattice, extending it in a specialised direction or using the lattice as a merging tool for multiple queries offer significant possibilities for IR. This approach can also be seen as a web mining one, in which the result of a step is exploited on the next step.

CreChainDo extends FCA for IR in a dynamical way. Lattices may evolve during the IR process. The user is not restricted anymore by a static and once computed structure.

References

1. C. Carpineto, A. Della Pietra, S. Mizzaro, and G. Romano. Mobile clustering engine. In *Proceedings of the 28th European Conference on Information Retrieval,*

ECIR 2006, volume 3936 of *LNCS*, pages 155–166. Springer, 2006.
2. C. Carpineto and G. Romano. Information retrieval through hybrid navigation of lattice representations. *International Journal of Computer Human Studies*, 45(5):553–578, 1996.
3. C. Carpineto and G. Romano. Effective Reformulation of Boolean Queries with Concept Lattices. In *Flexible Query Answering Systems, Third International Conference (FQAS'98)*, volume 1495 of *LNCS*, pages 83–94. Springer, 1998.
4. C. Carpineto and G. Romano. Order-Theoretical Ranking. *Journal of the American Society for Information Science*, 51(7):587–601, 2000.
5. C. Carpineto and G. Romano. Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. *Journal of Universal Computer Science*, 10(8):985–1013, 2004.
6. R.J. Cole, P.W. Eklund, and G. Stumme. Document retrieval for email search and discovery using formal concept analysis. *Journal of Applied Artificial Intelligence (AAI)*, 17(3):257–280, 2003.
7. P. A. Dmitriev, N. Eiron, M. Fontoura, and E. Shekita. Using annotations in enterprise search. In *Proceedings of the 15th International Conference on World Wide Web*, pages 811–817, 2006.
8. J. Ducrou. DVDSleuth: A Case Study in Applied Formal Concept Analysis for Navigating Web Catalogs. In *Conceptual Structures: Knowledge Architectures for Smart Applications, 15th International Conference on Conceptual Structures (ICCS 2007)*, volume 4604 of *LNCS*, pages 496–500. Springer, 2007.
9. J. Ducrou, B. Vormbrock, and P. W. Eklund. FCA-Based Browsing and Searching of a Collection of Images. In *Conceptual Structures: Inspiration and Application, 14th International Conference on Conceptual Structures (ICCS 2006)*, volume 4068 of *LNCS*, pages 203–214. Springer, 2006.
10. S. T. Dumais, E. Cutrell, R. Sarin, and E. Horvitz. Implicit queries (iq) for contextualized search (demo description). In *Proceedings of SIGIR 2004*, page 594, 2004.
11. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin, 1999.
12. R. Kassab, J.-C. Lamirel, and E. Nauer. Novelty detection for modeling user's profile. In *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 830–831, California, 2005. AAAI Press.
13. D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.
14. S. Y. Rieh and H. Xie. Analysis of multiple query reformulations on the web: the interactive information retrieval context. *Information Processing and Management*, 42(3):751–768, 2006.
15. J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323, 1971.
16. G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297, 1990.
17. X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of SIGIR'05*, 2005.
18. Snowball project web site. http://snowball.tartarus.org/, visited june 2007.
19. L. Szathmary and A. Napoli. CORON: A Framework for Levelwise Itemset Mining Algorithms. In *Supplementary Proceedings of The Third International Conference on Formal Concept Analysis (ICFCA '05), Lens, France*, pages 110–113, 2005.
20. C. J. van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

Use of Model Driven Engineering in Building Generic FCA/RCA Tools [⋆]

J.-R. Falleri[1], G. Arévalo[2], M. Huchard[1], and C. Nebut[1]

[1] LIRMM, CNRS and Université de Montpellier 2,
161, rue Ada, 34392 Montpellier cedex 5, France
{falleri, huchard, nebut}@lirmm.fr
[2] LIFIA – Facultad de Informatica (UNLP)
La Plata, Argentina
garevalo@sol.info.unlp.edu.ar

Abstract. Within maintenance software methodologies that analyze existing applications, Relational Concept Analysis (RCA) is an efficient approach to build abstractions in any language, using the existing relations between different software artifacts. Nowadays, there are several RCA-based tools, where a critical aspect is the lack of genericity in the FCA mapping to translate input (and output) data to (and from) formal/relational contexts. Most of the tools provide specific translators that can be used only with the analyzed application domain and their code needs to be changed when the framework of analysis evolves. Using Model-Driven Engineering, we propose a generic encoding/decoding process, which performs this translation with only the configuration of an encoder/decoder tool. This approach eases the integration of a FCA/RCA process in a tool and facilitates its usage on a wide range of input data formats.

1 Introduction

Within maintenance software methodologies that analyze existing applications, Relational Concept Analysis (RCA) [1] is an efficient approach to build abstractions in any language, such as modeling languages (UML) or programming languages (Java), that provide specialization-generalization mechanism between different software entities. Abstractions are built for entities using the existing relations between them. Nowadays, there are several tools implementing the RCA process. Starting from contexts and relational contexts, these tools automatically build a set of lattices (called Concept Lattice Family (CLF)) that contains new abstractions.

However, when applied in real case studies, these tools must cope with two main difficult tasks. The first task is the encoding of the model or program to be restructured using a Relational Context Family (RCF), and the second task is the decoding back of the obtained Concept Lattice Family (CLF) into the initial language. For example, several existing methodologies, such as [2,3], propose tools

⋆ France Télécom R&D has supported this work

to encode a UML model into a RCF, then apply a RCA technique, and finally decode back the CLF into a UML model. Note that the use of encoder/decoder is a common problem to FCA and RCA: For each type of model or program one may want to restructure with RCA, a new specific encoder/decoder is needed. To cope with this problem, this paper presents a generic encoder/decoder, whose configuration is based on the structure of the type of model or language to be restructured. This means that, given the elements of the input model or language and the elements that cause the abstractions' building, the repetitive and difficult work in creating RCF and then decoding CLF is fully automatic. For example, in a UML model we will create abstractions for classes and associations, based on the name of the roles of the associations and the name of the classes, whereas in Java we will create abstractions only for classes.

We achieve the design and configuration of the generic encoder/decoder through the use of the Model-Driven Engineering (MDE) paradigm [4], in which every used or produced artifact during the software development is a model. The structure of the model is defined with a metamodel. We proceed in two steps to solve the problem of the generic encoder/decoder. First, we have implemented the RCA process in a MDE-oriented way [2], defining metamodels for RCF and CLF. Second, we have defined a metamodel to define models of the encoding/decoding configuration corresponding to a given language to restructure. To restructure a model, one just has to decide which elements of the model are useful for the RCA-based restructuration, filling up a configuration model. In this paper we focus on the configuration step, i.e. the way we provide a generic encoder/decoder for a RCA-based restructuration.

The rest of the paper is organized as follows. Section 2 first briefly introduces FCA and RCA, and analyses the existing FCA and RCA tools (in particular w.r.t. their capacity to offer generic inputs and outputs). Then, Section 3 gives an overview of Model-Driven Engineering and explains the reasons of the choice of such a paradigm. Then, Section 4 details the mechanisms involved in the generic encoder/decoder for RCA-based model restructuration and illustrates them with UML and Java. Last, Section 5 concludes and gives feedback on the use of MDE for RCA tools.

2 Formal and Relational Concept Analysis

2.1 Background and Definitions

Formal concept analysis (FCA) [5,6,7] is a branch of lattice theory that allows us to identify meaningful groupings of objects that have common attributes.

Definition 1 (Formal Context). *A formal context is a 3-tuple $K = (O, A, I)$, where O and A are finite sets of objects and attributes respectively, and $I \subseteq O \times A$ an incidence relation where $\forall (o, a) \in I$, a is an attribute of the object o.*

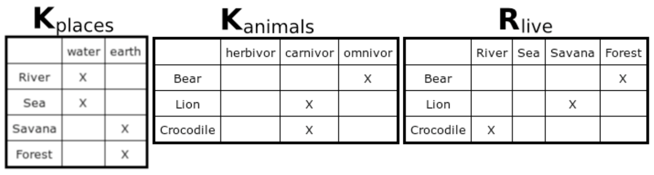
With a formal context, several concepts can be produced. A concept is a set of objects that share several attributes. It can be considered as an abstraction of these objects.

Definition 2 (Concept). *A concept is a pair (X, Y) with $X \subseteq O$, $Y \subseteq A$ and*
$X = \{o \in O | \forall y \in Y, (o, y) \in I\}$ *is the extent (covered objects),*
$Y = \{a \in A | \forall x \in X, (x, a) \in I\}$ *is the intent (shared attributes).*

In other words, the sets of objects and attributes are maximal, i.e. there is no other object that belongs to the concept extent and owns all the attributes of the intent. Moreover, there is no other attribute that belongs to the concept intent and that is owned by all the objects of the extent. These concepts can be organized in a lattice: A concept c_1 is lower than a concept c_2 if the extent of c_1 is included in the extent of c_2 (and inversely, the intent of c_2 is included in the intent of c_1).

FCA is efficient when considering objects described by binary attributes. In order to deal with non binary attributes, FCA has to be extended. One interesting possible extension is to classify several objects taking into account the relations between them. Among the existing approaches, Relational Concept Analysis [1] takes into account this possibility and considers the links between the objects as relational attributes. At the beginning, these relational attributes connect an object to several objects. Then, they connect an object to several concepts that have emerged from the classification of the objects. Instead of having just one formal context, RCA needs to define several formal contexts. Several relational contexts, that represent relations between objects defined in the formal contexts, can be added. Circularity between relations is admitted. Those formal and relational contexts together are called a Relational Context Family (RCF). Figure 1 shows a simple RCF, that deals with animals living in different places. There is a formal context which describes the animals ($K_{animals}$), another one which describes the places (K_{places}), and finally a relational context (R_{live}) describing the living relation between the animals and the places.

Definition 3 (Relational Context Family). *A Relational Concept Family $\mathcal{R}$ is a pair (K, R). K is a set of formal contexts $K_i = (O_i, A_i, I_i)$, R is a set of relational contexts $R_j = (O_k, O_l, I_j)$ (O_k and O_l are object sets of contexts K_k and K_l).*


Kplaces

	water	earth
River	X	
Sea	X	
Savana		X
Forest		X

Kanimals

	herbivor	carnivor	omnivor
Bear			X
Lion		X	
Crocodile		X	

Rlive

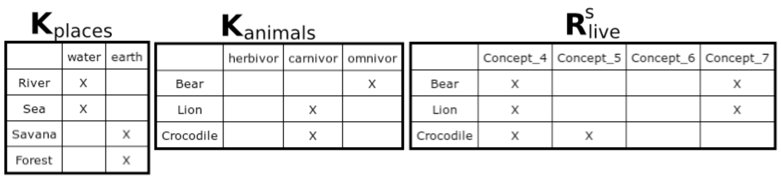
	River	Sea	Savana	Forest
Bear				X
Lion			X	
Crocodile	X			

Fig. 1. A relational context family

New abstractions emerge from a RCF by iterative building of concept lattices, and enhancement of the RCF with the concepts discovered in the concept lattices. The steps of this iterative construction are described below.

Initialization step. The created lattices at this step are the same as the one that would have been created using classical FCA: for each formal context K_i, a lattice $\mathcal{L}_i^0$ is created. Figure 3 (left) shows the initial lattices corresponding to the contexts of the Figure 1.

Step n+1. For each relational context $R_j = (O_k, O_l, I_j)$, a context $R_j^s = (O_k, A, I)$ is created. A contains the extents of the concepts of the lattice $\mathcal{L}_l^n$, and the incidence function I contains the element (o, a) if $S(R(o), a)$ is $true$. The function S is called a scaling operator. The most common scaling operators are $S_\exists(R(o), a)$, that is $true$ iff $\exists x \in R(o), x \in a$, and $S_\forall(R(o), a)$, that is $true$ iff $\forall x \in R(o), x \in a$. Figure 2 shows such an updated relational context (we used the $S_\exists$ scaling operator). It connects animals with concepts of the lattice (shown in Figure 3 (left)) corresponding to K_{places}. In this lattice, the concept 7 is an abstraction of the places on earth. The application of FCA on $K_k \cup \{R_j^s = (O_k, A, I)\}$ creates new concepts that are added to $\mathcal{L}_k^n$ to obtain $\mathcal{L}_k^{n+1}$. Figure 3 (right) shows such an updated lattice family. $\mathcal{L}_{animals}$ has been calculated using $K_{animals} \cup R_{live}^s$. The concept 10 of $\mathcal{L}_{animals}$ has been created using relational descriptions. It represents the animals that live on earth.



$\mathbf{K}_{places}$

	water	earth
River	X	
Sea	X	
Savana		X
Forest		X

$\mathbf{K}_{animals}$

	herbivor	carnivor	omnivor
Bear			X
Lion		X	
Crocodile		X	

$\mathbf{R}_{live}^{s}$

	Concept_4	Concept_5	Concept_6	Concept_7
Bear	X			X
Lion	X			X
Crocodile	X	X		

Fig. 2. RCF of Fig. 1 after the first step

This process stops when all the lattices of a step n are isomorphic to those of the step $n - 1$. After this step, the relational contexts will not be modified and no more concepts will emerge.

2.2 Encoding and decoding input data in FCA/RCA-based tools

In order to understand the existing limitations of FCA/RCA-based tools, an overview of the existing ones is needed taking into account their encoding/decoding operation. This operation, named *FCA mapping*, consists in converting input data into FCA contexts and then converting the built lattices back to the initial format. Most of the existing FCA-based tools [8,9] do not consider the FCA mapping as a crucial task. However, if a tool builder wants to integrate any FCA-based tool, he has to build himself a mapping tool to convert the input data to be analyzed into the input format of the FCA-based tool. Such an approach is expensive, moreover the knowledge of the configuration is included in the mapping tool, and the code of this tool has to be modified if the configuration changes.

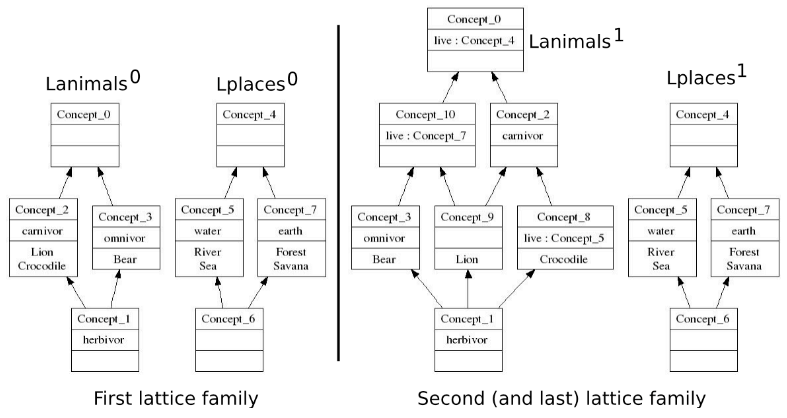


Fig. 3. The lattice families corresponding to the RCF of Fig. 1 and Fig. 2

Some FCA-based tools like *ConAn* [10](a FCA-based reengineering tool) have identified the importance of defining FCA mappings, and the advantages and drawbacks respectively. *ConAn* uses a software model expressed in a language independent way (called a FAMIX model), and FCA mappings to build a formal context. After building the corresponding lattices, *ConAn* builds a new model, called a high-level view (to help in the software analysis) in a format understandable by a regular developer with no lattice theory knowledge. Unfortunately, *ConAn* is focused on software reengineering and does not provide facilities to express neither FCA mappings nor high-level view formats. It is still necessary to hand-code a translator between a base model (named as FAMIX) and a formal context and then, between the concept lattices and the high-level view.

From our viewpoint, the most advanced approaches are [3] and [11]. They aim at discovering new abstractions in UML models using RCA. They use different tools: (1) a CASE tool (Objecteering [12] for [3], Eclipse [13] for [11]) to generate a context family, (2) *Galicia* [14] to build the corresponding lattice family, and then, (3) the CASE tool again to build an improved UML model. The FCA mapping between a UML model and a context family can be configured in both tools. However, this configuration mechanism is specific to UML. Therefore these tools cannot be used to analyze other input data, such as Java source code.

A FCA-based tool to easily define FCA mappings on a wide range of input data formats (UML model, Java code, Smalltalk code, . . .) is still missing. The design of such a tool is the contribution of this paper. To realize it, we need to introduce a high-level configuration mechanism which allows to express which elements of the input data have to be transformed into FCA items. For that purpose, it is necessary to use a high-level modeling language, which is able to represent a UML model or a Java source code in a common way. Such support is provided by the Model Driven Engineering paradigm, described in the next section.

3 Model Driven Engineering

Model Driven Engineering [4] is a recent software development paradigm. It was introduced to deal more with abstractions rather than code. In a MDE-based development, every produced or used artifact (including code) is a model, whose structure is defined by a metamodel (a model is said to conform to a metamodel). To pragmatically handle two models that conform to two different metamodels (for example to transform a UML model into a Relational Database model), a program has to be written, dealing with both metamodels. For that purpose, MDE assumes the existence of a unique meta-metamodel. Such a meta-metamodel allows to define how a metamodel is structured. Mainly, two meta-metamodels are used: EMOF [15] (defined by the OMG) and Ecore [16], (defined by Eclipse). Since we have built our tool with the Eclipse platform, we have chosen the meta-metamodel Ecore. However, Ecore and EMOF have no significant differences. In the following, we will use Ecore. The meta-metamodel is the last level in the modeling hierarchy (shown in Figure 4), and is expressive enough to describe itself. Therefore a meta-meta-metamodel is not necessary.

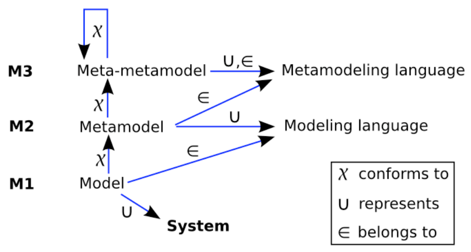


Fig. 4. The metamodeling hierarchy

A fundamental element of MDE is the notion of *model transformation*. A model transformation is a program or a set of rules that takes one or several models in input (conform to one or several metamodels) and produces as output one or several models.

In Section 1, we explained that the goal of this paper is to provide a high-level configuration mechanism allowing to define FCA mappings. Such a mechanism must be able to define FCA mappings, for example, from a UML model as well as from Java code, because Ecore can describe either a UML model or Java code. Therefore, a FCA mapping can be considered as a model transformation based on Ecore (and thus handling all kinds of models) from a model into a RCF model, and reciprocally from a CLF model to an output model (we will introduce what is a RCF and CLF model in the next section). Since all the information from a UML model or Java code is not relevant for a RCA restructuration, the tool builder still have to be able to select what kind of data he wants to place in the FCA items, that will be produced by the transformation. The next section explains our approach in detail, based on MDE, that allows an easy definition of FCA mappings.

4 FCA/RCA with a MDE approach

A global view of a FCA/RCA process using a MDE approach is shown in Figure 5. The purpose of using such a process on an input data (called model in the MDE approach) is to lead to the creation of output data that contains relevant abstractions. In this process, we consider that input and output models conform to the same metamodel. This is the most common case when using a RCA process for restructuration or abstraction discovery.

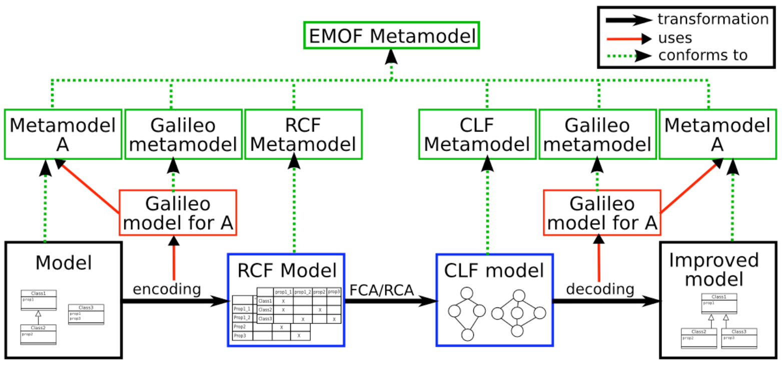
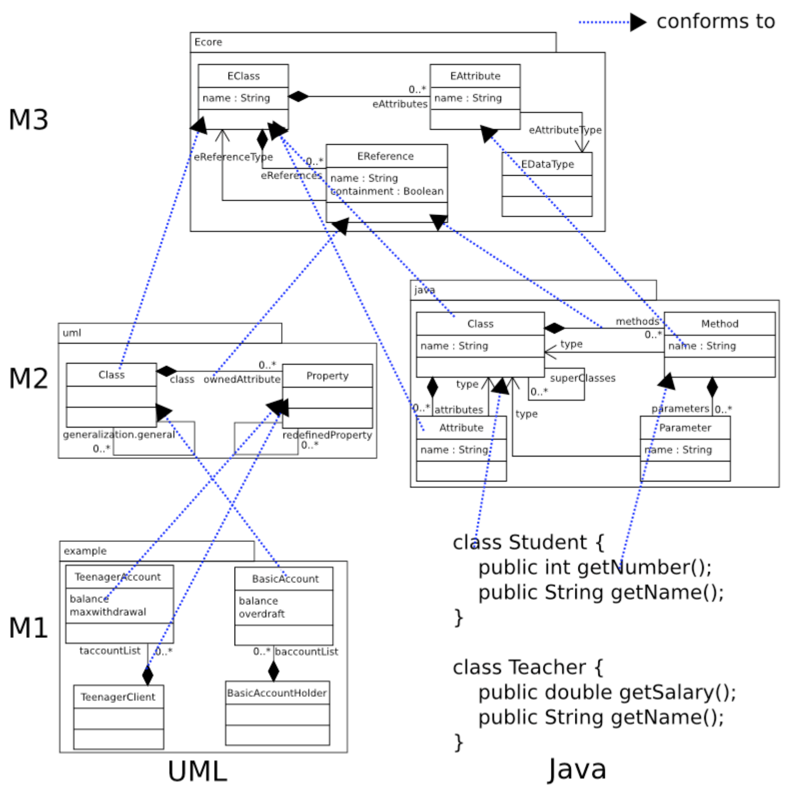


Fig. 5. Process overview

The *encoding* transformation of the process aims at producing FCA/RCA items from an input model. Only some configuration data is required in order to perform that transformation. The FCA/RCA transformation (described in section 2) builds the lattices corresponding to the input contexts. The *decoding* transformation consists in converting the produced lattices back to output data, conforming to the same metamodel as that before the *encoding* transformation. Since the input and output data metamodels are the same, the *decoding* transformation uses the same configuration as the *encoding* one to produce the output model.

To illustrate how the *decoding* and the *encoding* steps work, we use two sample input models: a UML model and a Java source code. Figure 6 shows these models in a MDE fashion, detailing models, metamodels and the meta-metamodel. The *conforms to* arrows show the connections between the elements of the models and the elements of the metamodel; and between the metamodels and the meta-metamodel. For the sake of clarity, the Java, UML and Ecore metamodels have been presented in a reduced form. Next we describe how these models are transformed through the *encoding* and *decoding* operations.



```
class Student {
    public int getNumber();
    public String getName();
}

class Teacher {
    public double getSalary();
    public String getName();
}
```

Fig. 6. The sample models

To remain in a MDE approach, a RCF metamodel was created, which is is able to define data structures similar to the one described in Section 2. Figure 7 shows this metamodel: An RCF is composed of several formal contexts (called *EntityAttributeContext* in the metamodel) and several relational contexts (called *InterEntityContext*). These contexts contain an incidence relation (represented as a set of *Pair* in the metamodel).

We use an UML and a Java model. In this restricted example, we are interested in using RCA on the UML model to create new superclasses by factorizing the properties of the existing classes, based on their names. For Java, we want to use RCA in order to create new superclasses that factorize methods from the existing classes, based on their names. To achieve these goals, the *encoding* operation has to generate formal and relational contexts describing the previously quoted elements. We introduce a configuration metamodel to give this information to the transformation.

Figure 8 shows this metamodel. Models conform to this configuration metamodel are used in the *encoding* transformation to dynamically choose the elements to consider and how to encode them. Figure 9 shows configuration models

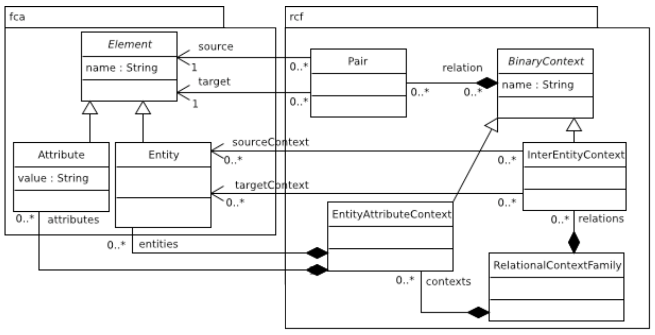


Fig. 7. The Relational Context Family metamodel

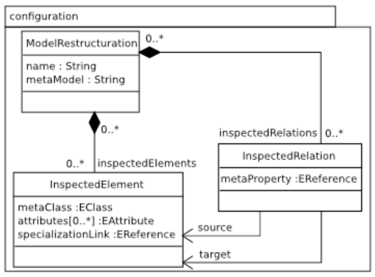


Fig. 8. The encoding/decoding configuration metamodel

for UML and Java that lead to the creation of contexts describing the elements we want to analyze.

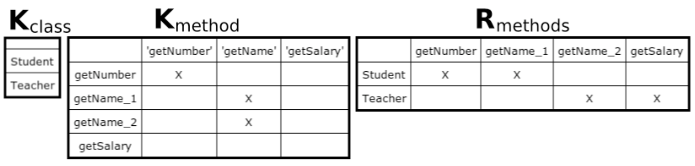
The *encoding* transformation works as follows. First, several sets of elements lead to the creation of several formal contexts, one for each kind of element. The *InspectedElement* elements coming from the configuration model parameterize this stage of the transformation. For each *InspectedElement*, a set of elements is dynamically selected via the *metaClass* information. The binary attributes that describe this set of elements are selected via the *attributes* information. A formal context is finally created using these elements as objects and the binary attributes. Given the configurations plotted by Figure 9, two formal contexts are created for the UML model: one describing the classes and the other one, the properties (using the *name* as attribute). For the Java code, two formal contexts are created: one for the classes, the other one for the methods (using *name* as attribute). Figure 10 and 11 show the formal contexts generated from the examples.

Secondly, several relations between the elements of the input model are transformed into several relational contexts. The *InspectedRelation* elements coming from the configuration model parameterize this stage of the transformation. For each *InspectedRelation*, a relation is selected in the input model via the *metaProperty* information. The *source* and *target* are used to select the elements

```
Config. for UML Class Model restructuring:

Inspected Elements:
Class: attributes = [],
 specializationLink = "generalization.general"
Property: attributes = ["name"],
 specializationLink = "redefinedProperty"

Inspected Relations:
ownedAttribute: source = Class,
 target = Property
type: source = Property,
 target = Class
```

```
Config. for Java source code restructuration:

Inspected Elements:
Class: attributes = [],
 specializationLink = "generalization.general"
Method: attribute = ["name"]

Inspected Relations:
methods: source = Class, target = Method
```

Fig. 9. Two configuration models



K_{class}

Student
Teacher

K_{method}

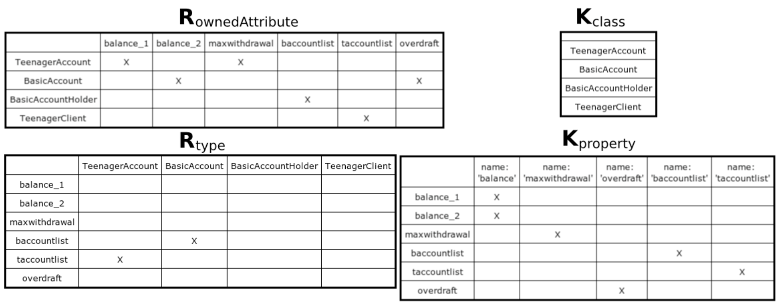
	'getNumber'	'getName'	'getSalary'
getNumber	X		
getName_1		X	
getName_2		X	
getSalary			

$R_{methods}$

	getNumber	getName_1	getName_2	getSalary
Student	X	X		
Teacher			X	X

Fig. 10. The generated Java contexts

(described in the previously built formal contexts) involved in this relational context. With the configuration in Figure 9, two relational contexts are created for the UML model: one for the *ownedAttribute* relation between the classes and the properties, and the other for the *type* relation between the properties and the classes. For the Java code, one relational context is created to describe the *methods* relation between the classes and the methods. Figures 10 and 11 show relational contexts generated from the examples.



$R_{ownedAttribute}$

	balance_1	balance_2	maxwithdrawal	baccountlist	taccountlist	overdraft
TeenagerAccount	X		X			
BasicAccount		X				X
BasicAccountHolder				X		
TeenagerClient					X	

K_{class}

TeenagerAccount
BasicAccount
BasicAccountHolder
TeenagerClient

R_{type}

	TeenagerAccount	BasicAccount	BasicAccountHolder	TeenagerClient
balance_1				
balance_2				
maxwithdrawal				
baccountlist		X		
taccountlist	X			
overdraft				

$K_{property}$

	name: 'balance'	name: 'maxwithdrawal'	name: 'overdraft'	name: 'baccountlist'	name: 'taccountlist'
balance_1	X				
balance_2	X				
maxwithdrawal		X			
baccountlist				X	
taccountlist					X
overdraft			X		

Fig. 11. The generated UML contexts

The *decoding* transformation aims at producing a model using the concept lattices produced with the RCA. Figure 12 shows the metamodel of a concept lattice family. This transformation uses the same configuration model as the *decoding* transformation. The input is a family of concept lattices, one for each context of the input RCF. These contexts are used to represent a kind of objects present in the source model. For the UML model, there is one lattice for the classes and one for the properties. For the Java code, there is one lattice for the classes and one for the methods. Therefore, the elements built from a lattice have the same metaclass as the elements that lead to the creation of this lattice. For instance with UML, the class lattice coming from the class formal context produces classes in the ouput UML model. The partial order between the concepts of a lattice is used to create relations between the elements generated from the concepts of this lattice. The name of the relation to create is in the *specializationLink* information from the *InspectedElement* coming from the configuration model associated with the lattice. With UML, the partial order between the concepts of the class lattice is interpreted as a *generalization.general* between the classes. Binary attributes and relational attributes from a concept of a lattice need to be placed in the corresponding elements. The binary attributes lead to the creation of attributes in the corresponding element. Relational attributes generate relations in the output model between the element corresponding to the concept and the element corresponding to the target concept of the relational attribute.

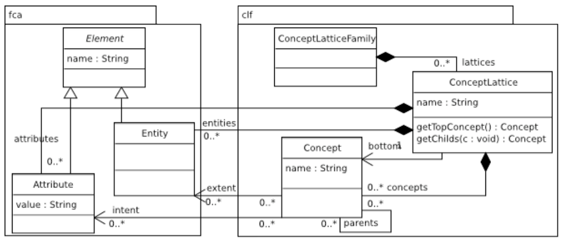


Fig. 12. The Concept Lattice Family metamodel

5 Conclusions and Future Work

We have proposed a generic way to encode models and programs so that they can easily benefit from a restructuration based on FCA or RCA, as well as the reciprocal generic way to decode the obtained lattices. The genericity of the approach has been designed using the Model-Driven Engineering paradigm, and is based on the knowledge of the metamodels of the artifacts to restructure and the underlying common meta-metamodel. To restructure a model, one has just to give (in addition to the model itself): the metamodel of the model, and a configuration model making precise which kinds of elements of the input model are to be taken into account. This approach is implemented in a tool written in Java EMF that has been experimented with UML models (from France Télécom projects [17]), Java programs, OWL, and Ecore models.

MDE gave us further significant benefits. The parameters of the RCA process have been identified and properly modeled, consequently the RCA restructuration can be easily fine-tuned. We plan to go a step further in the parameterization of the RCA process, allowing the user to choose the algorithm to build lattices and the scaling operator, or to define the number of steps he wants the algorithm to perform.

Further work will consist in adapting our approach so that the input metamodel used for the encoding can be different from the output metamodel used for the decoding. This is useful when the discovered abstractions cannot be expressed with the input metamodel (but could be expressed with a similar metamodel). For example, restructuring Java programs may create multiple class inheritance, that cannot be represented with a Java program, but that can be represented with a UML class model.

References

1. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. **49**(1-4) (2007) 39–76
2. Arévalo, G., Falleri, J.R., Huchard, M., Nebut, C.: Building Abstractions in Class Models: Formal Concept Analysis in a Model-Driven Approach. In Nierstrasz, O., Whittle, J., Harel, D., Reggio, G., eds.: MoDELS. Volume 4199 of Lecture Notes in Computer Science., Springer (2006) 513–527
3. Dao, M., Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Towards Practical Tools for Mining Abstractions in UML Models. In Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J., eds.: ICEIS (3). (2006) 276–283
4. Kent, S.: Model Driven Engineering. In Butler, M.J., Petre, L., Sere, K., eds.: IFM. Volume 2335 of Lecture Notes in Computer Science., Springer (2002) 286–298
5. Birkhoff, G.: Lattice theory. New York (1948) 247
6. Barbut, M., Monjardet, B.: Ordre et Classification: Algèbre et Combinatoire. Volume 2. Hachette (1970)
7. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1999)
8. Conexp: Conexp. http://conexp.sourceforge.net/ (2006)
9. Burmeister, P.: Formal Concept Analysis with ConImp: Introduction to the Basic Features. Fachbereich Mathematik, Technische Universität Darmstadt (2003)
10. Arévalo, G., Ducasse, S., Nierstrasz, O.: Lessons Learned in Applying Formal Concept Analysis to Reverse Engineering. In Ganter, B., Godin, R., eds.: ICFCA. Volume 3403 of Lecture Notes in Computer Science., Springer (2005) 95–112
11. Seuring, P.: Design and implementation of a UML model refactoring tool. Master's thesis, Hasso-Plattner-Institute for Software Systems Engineering at the University of Postdam (2005) http://www.lirmm.fr/˜huchard/Documents/Papiers/PhilippSeuringMasterThesis.pdf.
12. Softeam: Objecteering. http://www.objecteering.com/ (2007)
13. Eclipse: Eclipse. http://www.eclipse.org/ (2007)
14. Galicia: Galicia. http://www.iro.umontreal.ca/ galicia/ (2005)
15. OMG: MOF 2.0 core specification. http://www.omg.org/cgi-bin/doc?ptc/2004-10-15 (2004)
16. Eclipse: The Eclipse Modeling Framework (EMF) Overview. http://www.eclipse.org/emf/docs.php?doc=references/overview/EMF.html (2005)
17. Falleri, J.R., Arévalo, G., Huchard, M., Nebut, C.: Etude de l'extension d'une méthode de restructuration de diagrammes UML/ Tests sur la plate-forme Petra. Contrat particulier de Recherche Externalisée 5326. Tâche 2. LIRMM/France Télécom (2007)

Precalculating Component Interface Compatibility Using FCA

Gabriela Arévalo[1], Nicolas Desnos[2], Marianne Huchard[3], Christelle Urtado[2], and Sylvain Vauttier[2]

[1] LIFIA - Facultad de Informática (UNLP) - La Plata - Argentina
garevalo@sol.info.unlp.edu.ar
[2] LGI2P / Ecole des Mines d'Alès - Nîmes - France
{Nicolas.Desnos, Christelle.Urtado, Sylvain.Vauttier}@ema.fr
[3] LIRMM - CNRS and Univ. Montpellier 2 - Montpellier - France
Marianne.Huchard@lirmm.fr

Abstract. In component-based software engineering, software systems are built by assembling prefabricated reusable components. The compatibility between the assembled components is crucial. It is determined by the comparison of their exposed interfaces: required interfaces (describing the services the component needs) and provided interfaces (describing the services the other component offers) must match. Given a component, finding a compatible component in a component repository is not trivial. The idea of this paper is that organizing component directories with a yellow-page-like structure makes the search for suitable components more efficient. We propose a solution based on Formal Concept Analysis to precalculate a concept lattice to organize our components. It proves to be an efficient solution to both represent the component compatibility information and provide a browsable component organization to support the component search mechanism.

1 Introduction

The component-based approach is a recent and successful paradigm for software engineering, inspired by electronic engineering. In this approach, software systems are built by assembling prefabricated reusable components. The main objective is development cost reduction while maintaining high quality. Software components are externally described, as their electronic counterpart, by functionalities they support, and plugs which specify possible connections. In terms of the component-based software engineering (CBSE) domain, a component has required interfaces (needed services) and provided interfaces (offered services). Building a component assembly consists in connecting components in order to achieve a high-level functionality. Computation is then dispatched over the assembled components. Nevertheless, given a component repository, finding and connecting suitable components is not trivial because there is basically a need to determine service compatibility. Component repositories generally rely on component directories organized as white pages, that support component search by name. Such an organization is not optimal because it leaves the burden of finding suitable components to the assembly phase.

Previous work on component assemblies [1, 2] made us realize how critical the component-search mechanism was for such software engineering processes as component assembly building or evolution. The idea of this paper is that organizing component directories with a yellow-page-like structure makes the search for suitable components more efficient. Our approach is based on Formal Concept Analysis (FCA), where precalculating a lattice of service compatibilities is a means to organize the component directory so as to both explicitly represent relationships among components and efficiently search for suitable components when needed. Having a user-readable component structure that shows component compatibility is a plus and separating the concerns of compatibility calculus and assembly is more rational. Such an organized component directory can then be used to find a component that can assemble to (search for a compatible component) or replace (search for a substitutable component) a selected one. It also enables the discovery of new abstract components having a higher degree of reusability which enriches the component directory.

The remainder of this paper is organized as follows. Using a small sale system example, Section 2 shows how reasoning about component compatibility is mainly based on functionality signatures. Section 3 describes an extension of object-oriented type theory for functionality signature comparison. Then, after recalling chosen basics of FCA, Section 4 shows the building of a lattice of functionality signatures using the point of view of replacing a required functionality by another. Section 5 presents several uses of the proposed lattice, Section 6 compares our approach to related work and Section 7 mentions future work.

2 Component Compatibility: a Sale System Example

Service and functionality compatibility, as an *a priori* requirement for component compatibility, is discussed based on elements of a sale management application. Figure 1 shows several connection situations. When a component (*e.g.,* `ChildOrder`) is introduced in an assembly to satisfy a service (*e.g.,* enabling children to order products), the required interfaces involved in the same collaboration (here, `CustomerCreation`) have to be connected in the assembly. This may demand finding a compatible component either among those of the assembly, or in a component repository.

Component compatibility is generally defined as the syntactic comparison between the components' interfaces, which more precisely consists in comparing pairs of functionality signatures from these interfaces. To easily understand the discussion, the reader can draw a parallel with the ordinary function-call mechanism: required functionalities can be seen as function calls while provided functionalities are function definitions. Functionalities are described by their signatures, *i.e.,* their name, parameter type list and return type. To be run, a function needs its call to contain the expected information, as declared in its signature. The position, where the call appears in the code, expects in return a resulting

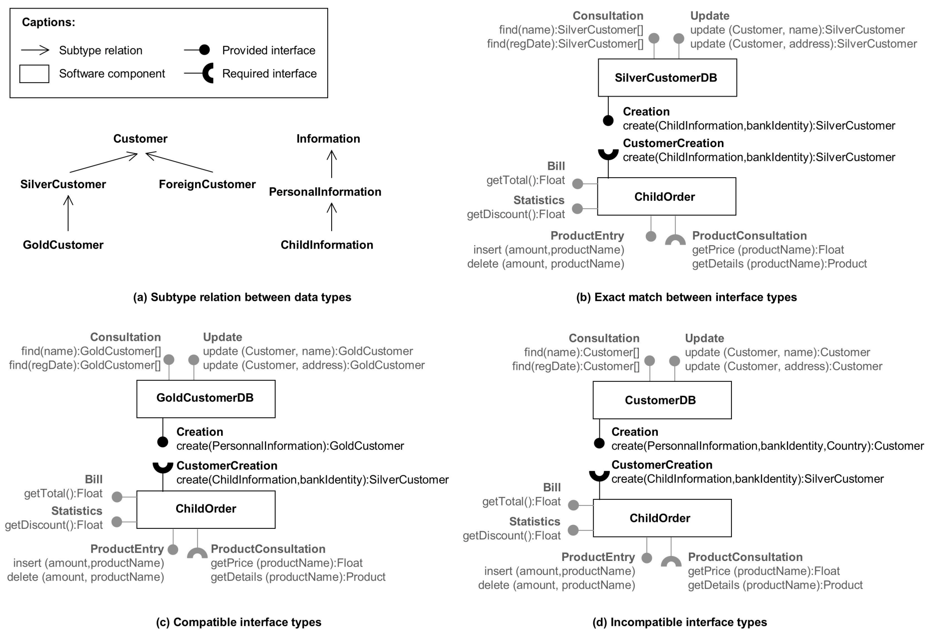

Captions:
Subtype relation
Software component
Provided interface
Required interface
Customer
SilverCustomer
ForeignCustomer
GoldCustomer
Information
PersonalInformation
ChildInformation
(a) Subtype relation between data types
Consultation
find(name):SilverCustomer[]
find(regDate):SilverCustomer[]
Update
update (Customer, name):SilverCustomer
update (Customer, address):SilverCustomer
SilverCustomerDB
Creation
create(ChildInformation,bankIdentity):SilverCustomer
CustomerCreation
create(ChildInformation,bankIdentity):SilverCustomer
Bill
getTotal():Float
Statistics
getDiscount():Float
ChildOrder
ProductEntry
insert (amount,productName)
delete (amount, productName)
ProductConsultation
getPrice (productName):Float
getDetails (productName):Product
(b) Exact match between interface types
Consultation
find(name):GoldCustomer[]
find(regDate):GoldCustomer[]
Update
update (Customer, name):GoldCustomer
update (Customer, address):GoldCustomer
GoldCustomerDB
Creation
create(PersonnalInformation):GoldCustomer
CustomerCreation
create(ChildInformation,bankIdentity):SilverCustomer
Bill
getTotal():Float
Statistics
getDiscount():Float
ChildOrder
ProductEntry
insert (amount,productName)
delete (amount, productName)
ProductConsultation
getPrice (productName):Float
getDetails (productName):Product
(c) Compatible interface types
Consultation
find(name):Customer[]
find(regDate):Customer[]
Update
update (Customer, name):Customer
update (Customer, address):Customer
CustomerDB
Creation
create(PersonnalInformation,bankIdentity,Country):Customer
CustomerCreation
create(ChildInformation,bankIdentity):SilverCustomer
Bill
getTotal():Float
Statistics
getDiscount():Float
ChildOrder
ProductEntry
insert (amount,productName)
delete (amount, productName)
ProductConsultation
getPrice (productName):Float
getDetails (productName):Product
(d) Incompatible interface types

Fig. 1. Type ordering information and three possible assembly situations

data corresponding to the declared return type of the function. Figure 1(b) illustrates the simplest assembly case: the `SilverCustomerDB` component provides an interface which describes a `create` functionality that has exactly the same signature as the `create` functionality of the required `CustomerCreation` interface of the `ChildOrder` component – that is the same parameter type list and the same return type. However, exact signature match is not always possible: the component repository may only contain components with close capacities. In Figure 1(c), the `GoldCustomerDB` component offers a close provided creation functionality which turns out to be compatible with the one required by the `ChildOrder` component. There are three differences between the two signatures:

- The provided `create` functionality needs information about a person. As shown on Figure 1(a), `ChildInformation` is a subtype of `PersonnalInformation`. This means that a child is described by more information than a person. As a consequence, the required functionality is able to call the provided functionality as the call contains more information than needed by the provided functionality to run.
- The provided `create` functionality only needs information about a person: it has no other parameter. The fact that the required `create` gives extra information (an extra `bankIdentity` parameter is passed to the functionality call) does not hamper the assembly.
- The provided `create` functionality returns a parameter of type `GoldCustomer`. As shown in Figure 1(a), `GoldCustomer` is a subtype of `SilverCustomer`.

As in the other cases, the required `create` can still call the provided functionality as it will receive in return more information than needed.

In these three cases, the extra information provided during the call or the return phases can be ignored: the components can be assembled. Figure 1(d) presents a case of functionality incompatibility. The `CustomerDB` component provides a `create` functionality that is not compatible with the `create` functionality of the `ChildOrder` component. Two signature differences make the assembly impossible:

– The provided `create` needs information about the country of the customer: the required `create` is unable to provide such information.
– The provided `create` returns a parameter of type `Customer`. As shown in Figure 1(a), `Customer` is a supertype of `SilverCustomer`. The required `create` therefore receives in return an object which is too general and contains less information than expected.

This small example illustrates the need for signature comparison based on a model of component compatibility, where parameter and return types have a great place. In realistic examples, an interface generally contains more than one functionality signature. This is why reasoning on component compatibility requires to have a theory of functionality signature compatibility, which can be extended to nesting (interface, component) levels. As shown in the example, signature compatibility is the result of the combination of subtyping orders on parameter and return types as well as presence / absence of parameters. In the case where the component assembler is human, mentally calculating this combination is rather difficult, increasing the errors when choosing a component among the available ones. Besides, a mental image, by the means of a classification of signatures, could be quite useful. At the same time, this could give us keys for helper tools that guide the human designer in its component choice, or even, in simple cases, that solve the choice problem automatically.

Thanks to its qualities for building classifications of entities, FCA is very useful to cope with this problem. The produced signature classifications can be used either to check compatibility if a component is available, to find a component (using the classification as an index for component access) or to build contingent interface or component classifications by propagating specialization information. Thanks to FCA, signature classification is not reduced to the organization of existing signatures: new signatures emerge that are generalizations (abstractions) of the existing ones, giving the opportunity to the component developer to imagine new components that would be more reusable than the existing ones.

3 Basics of Static Types: Ordering Required and Provided Functionalities

This section focuses on presenting the domain knowledge about signatures that will be encoded into concept lattices in order to capture functionality substitutability information. Part of assembly soundness is based on static types, inspired by statically typed object-oriented languages [3]. In this context, static types are used to guide an efficient analysis that considerably limits dynamic type errors potentially provoked by the intensive usage of polymorphism. In object-oriented languages, polymorphic expressions can be bound to values of several types (classes in this context). The correctness of this binding relies on the possibility for an expression of a type to be seamlessly replaced (substituted) by any value of another type. This substitutability is possible under some sufficient conditions on types relative to their operations: the run-time type error that static analysis wants to avoid is the reception by an object of an operation request that it cannot deal with. In the object paradigm, when an operation is redefined in a subclass (subtype), type-safe definitions of operations respect two constraints:

– *Parameter type contravariance.* A parameter type has to vary in the opposite direction as subclassing: in the subclass, the parameter has a more general type (super-type) than in the superclass.
– *Return type covariance.* The return type has to vary in the same direction as subclassing: in the subclass, the return type has a more specialized type (subtype) than in the superclass.

Let us study how this theory extends to the component domain. In order to define component substitutability, we only consider the smallest abstractions (functionalities) because the whole problem reduces to determining if a component that provides (*resp.* requires) a functionality can replace a connected component that provides (*resp.* requires) another functionality. Functionalities are described by their *name* (replacement is admitted only between same name functionalities), *parameter type list* (IN parameter types), and *return types* (OUT parameter types). Rules for substitutability are as follows for required functionalities (rules for provided functionalities are obtained by a symmetric reasoning):

– *IN parameter type specialization.* If a required functionality is able to send a parameter of some type, it can replace a required functionality which sends a parameter of a more general type (because the called functionality can ignore the information specific to the specialized type). In the example of Figure 2(a), required functionality `create(PersonalInformation)` can replace required functionality `create(Information)` because in the context where required `create(Information)` can be connected (namely, to provided `create(Information)`), required `create(PersonalInformation)` can also be connected as it can connect to provided `create(Information)` and to provided `create(PersonalInformation)` which is a superset.
This is an application of the substitutability rule in object-oriented languages and obeys the contravariance principle (because in the provided point of view, IN parameter types need generalization).
– *IN parameter addition.* If a required functionality is able to send a parameter of a particular type, it can replace a required functionality which does not send this kind of parameter (because the called functionality can ignore this

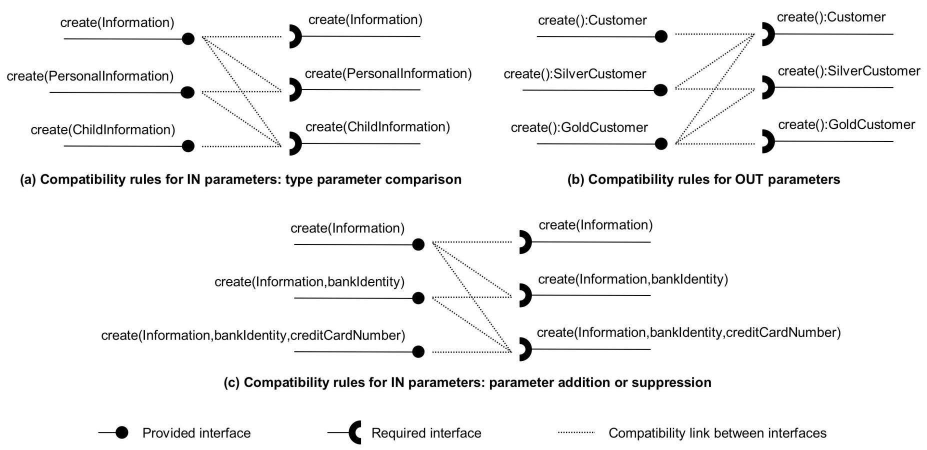

create(Information)
create(PersonalInformation)
create(ChildInformation)
(a) Compatibility rules for IN parameters: type parameter comparison
create():Customer
create():SilverCustomer
create():GoldCustomer
(b) Compatibility rules for OUT parameters
create(Information)
create(Information,bankIdentity)
create(Information,bankIdentity,creditCardNumber)
(c) Compatibility rules for IN parameters: parameter addition or suppression
Provided interface
Required interface
Compatibility link between interfaces

Fig. 2. Interface compatibility rules when parameter types and number vary

parameter). For example (Figure 2(c)), required `create(Information,bankIdentity)` can replace required `create(Information)` because in the context where required `create(Information)` can be connected, required `create(Information,bankIdentity)` can be connected too.

– *OUT parameter type specialization.* If a required functionality needs to receive some return type, it can replace a required functionality which needs to receive a more specialized type (as the extra information in the specific type can always be ignored). For example (Figure 2(b)), required `create():Customer` can replace required `create():SilverCustomer` because in the context where required `create():SilverCustomer` can be connected, required `create():Customer` can be connected too. This is another application of the substitutability rule in object-oriented languages and obeys the covariance principle (also considering the provided point of view).

4 Lattice of Functionality Compatibilities

This section firstly recalls the basics of FCA and then presents the building principles of the functionality signature lattice.

4.1 A Survival Kit for Formal Concept Analysis

The classification we build is based on the partially ordered structure known as *Galois lattice* [4] or *concept lattice* [5] which is induced by a context K, composed of a binary relation R over a pair of sets O (*objects*) and A (*attributes*) (Figure 3).

A concept C is a pair of corresponding sets X and Y such that:

$X = \{ \mathrm{x} \in O | \ \forall \mathrm{y} \in \mathrm{Y}, (\mathrm{x}, \mathrm{y}) \in R\}$ is called *extent* (covered objects)
$Y = \{ \mathrm{y} \in A | \ \forall \mathrm{x} \in \mathrm{X}, (\mathrm{x}, \mathrm{y}) \in R\}$ is called *intent* (shared features)

For example, $(12, bc)$[4] is a formal concept, but $(2, bc)$ is not. Establishing that $(12, bc)$ is a concept highlights the fact that objects 1 and 2 exactly share attributes b and c (and vice-versa). Furthermore, the set of all concepts $\mathcal{C}$ constitutes a lattice $\mathcal{L}$ when provided with the following specialization order based on intent / extent inclusion: $(X_1, Y_1) \leq_{\mathcal{L}} (X_2, Y_2) \Leftrightarrow X_1 \subseteq X_2$ (or equivalently $Y_2 \subseteq Y_1$). Figure 4 shows the Hasse diagram of $\leq_{\mathcal{L}}$.

	a	b	c	d	e	f	g	h
1		×	×	×	×			
2	×	×	×				×	×
3	×	×				×	×	×
4				×	×			
5			×	×				
6	×							×

Fig. 3. Binary relation of $K = (O, A, R)$ where $O=\{1, 2, 3, 4, 5, 6\}$ and $A=\{a, b, c, d, e, f, g, h\}$.

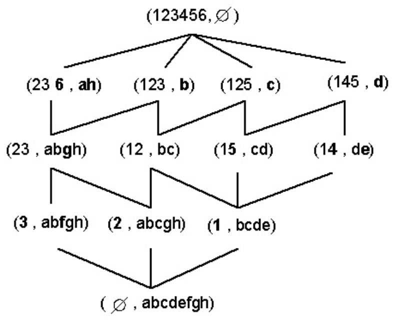


Fig. 4. Concept lattice $\mathcal{L}$.

4.2 Building the Lattice

We explain here the construction of the required functionality signature lattice. As provided functionality signatures are reversely ordered, the lattice we obtain can also be used to deal with them, when considered upside down.

We illustrate our explanation considering the required functionality `create (PersonalInformation,bankIdentity):SilverCustomer` and comparing it to `create (Information):GoldCustomer`. At first, for each functionality, attributes are deduced from IN and OUT parameter types that explicitly appear in the signature. These attributes are marked using $\times$ in Figure 5. Then, we infer attributes (marked with $\otimes$ in Figure 5) when their types are compatible, regarding specialization of signatures. Here are our inference rules:

- IN parameters. As explained previously, if a required functionality sends a parameter of some type, it also implicitly sends a parameter of any more general type (because the called provided functionality can ignore the part of the information that is specific to the specialized type).
- OUT parameters. If a required functionality expects to receive a return value of a type, any return value of a more specific type is also suitable (the received extra information can be ignored).

In component-based systems, all the information about the functionalities described in (provided or required) component interfaces is available at runtime.

[4] $(12, bc)$ is the compact notation for concept $\{1, 2\} \times \{b, c\}$.

Indeed, thanks to reflexivity (based on either metadata or introspection), a component can be asked each detail of one of its functionality signature (name of the functionality, types of IN and OUT parameters, etc.) at runtime. This capability could be used as the basis of an automatic process that would build the binary context and identify the inferred attributes.

Figure 6 depicts the lattice corresponding to the binary relation shown in Figure 5, built with the Galicia tool [6]. Concepts are presented using reduced intents and extents for readability sake: an object (signature) which belongs to the reduced extent of a concept is inherited by all concepts that are above (down-to-up); a property (IN or OUT parameter type) which belongs to the reduced intent of a concept is inherited by all concepts that are below (up-to-down).

The following section gives insights of the different operations this kind of lattice can support for the management of component directories.

	IN parameters								OUT param.			
	I	PI	CI	BI	CCN	ID	Co	A	C	SC	GC	FC
create(I):GC	×										×	
create(PI,BI):SC	⊗	×		×						×	⊗	
create(CI,BI,CCN):C	⊗	⊗	×	×	×				×	⊗	⊗	⊗
create(I,A,ID):C	×					×		×	×	⊗	⊗	⊗
create(I,BI,CCN,Co):FC	×			×	×		×					×

I	Information
PI	PersonalInfo.
CI	ChildInfo.
BI	BankIdentity.
CCN	CreditCardNb
Co	Country
ID	InitialDeposit
A	Address
C	Customer
SC	SilverCustomer
GC	GoldCustomer
FC	ForeignCustomer

Fig. 5. Encoding of a set of required functionality signatures

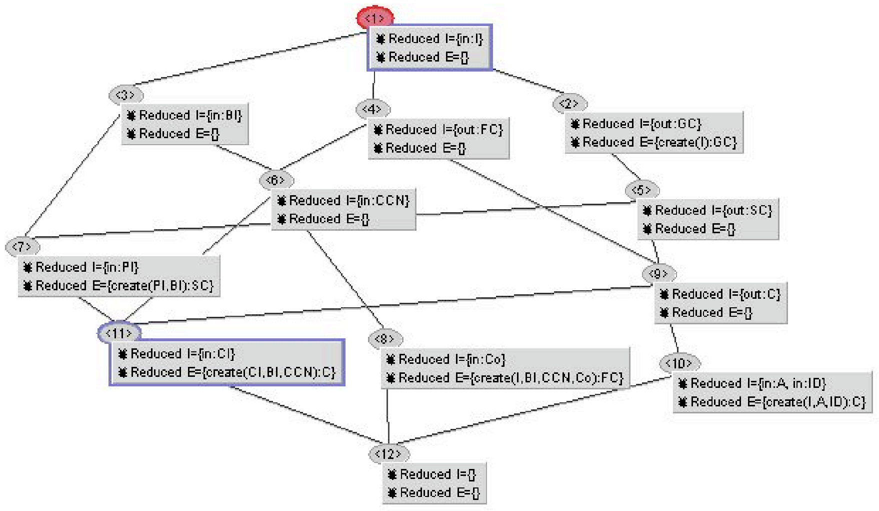


Fig. 6. Lattice $\mathcal{L}_s$ of service signatures

5 Using the Lattice of Functionality Compatibility

The lattice of signatures supports three main usages. Firstly, it determines which functionality can replace which other, in both cases of required or provided functionalities (Section 5.1). Secondly, the validity of the connection of a required functionality to a provided functionality can be stated (Section 5.2). Finally, in a prospective section, it is used as the starting point of a construction chain that goes from functionality signature to component type classification (Section 5.3).

5.1 Required/ Provided Functionality Substitution

Consider the lattice of Figure 6 with the viewpoint of required functionalities. In this lattice `create(I):GC` is represented by Concept 2 while `create(PI,BI):SC` is represented by Concept 7. Concept 2 is more general than Concept 7 which can be interpreted as Concept 7 can replace Concept 2. In a component architecture, a connection to a required functionality corresponding to Concept 2 can be replaced by a connection to a required functionality corresponding to Concept 7. In the general case, when there is a path between two concepts, the more specific one (which has more properties) can replace the more general one (which has a subset of properties) when the more general concept is connected (see Figure 7(a)).

The same lattice can also be used for provided functionality substitution if read in the reverse order (see Figure 7(b)). In the previous example, symmetric assertion can be made: in situations where provided functionality corresponding to Concept 7 are connected, provided functionality corresponding to Concept 2 can also be connected. Next rule formalizes how functionality substitution can generally be deduced from the lattice.

Substitution rule. Let C_{father}, C_{son} be two concepts of the functionality signature lattice such that $C_{son} \leq_{\mathcal{L}_s} C_{father}$. Functionalities of C_{son} can replace functionalities of C_{father} when the functionalities are required. Opposite replacement applies when the functionalities are provided.

5.2 Inferring Valid Connections

Both points of view (provided and required) can be combined to address the issue of component connection. Consider the signature `create(PI,BI):SC`. Required `create(PI,BI):SC` evidently can connect to provided `create(PI,BI):SC`. Given the substitution rule, provided functionalities which are upper in the lattice, for example provided `create(I):GC`, can be connected to required `create(PI,BI):SC` (see Figure 7(c)). Using the same rule in the symmetric case, required functionalities which are below in the lattice, for example required `create(CI,BI,CCN):C`, can be connected to provided `create(PI,BI):SC`. By transitivity, required `create(CI,BI,CCN):C` can be connected to provided `create(I):GC`. This is expressed in the following connection rule that formalizes how valid functionality connection can be deduced from the lattice.

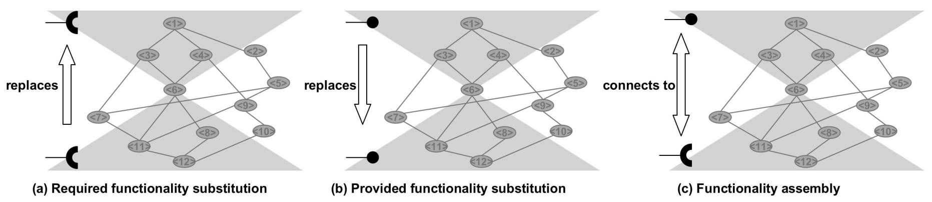

replaces
replaces
connects to
(a) Required functionality substitution
(b) Provided functionality substitution
(c) Functionality assembly

Fig. 7. Interpretation of the lattice of service signatures

Connection rule. Let C, C_{father}, C_{son} be three concepts of the functionality signature lattice such that $C_{son} \leq_{\mathcal{L}_s} C \leq_{\mathcal{L}_s} C_{father}$, required functionalities of C_{son} can be connected to provided functionalities of C_{father}.

5.3 Towards Component Classification

Although several component models exist for CBSE, most of them include the notions of interface and functionality. Components are reusable pieces of software that can be chosen off-the-shelf and have a high-level objective (database component, counter component, scheduler component, etc.). Interfaces group functionalities that form meaningful collaborations together and have a direction (provided or required).

Functionality classification is the starting point of a classification chain that ends in component classification, using a non-iterative version of Relational Concept Analysis [7]. We detail only the level of interfaces to give an idea of the process. An interface is described by a set of functionalities. Therefore, a formal context for interfaces naturally associates functionalities with interfaces. However, to benefit from knowledge acquired in the lattice of service signatures, a richer description of interfaces is obtained using the concepts of this lattice as formal attributes. For example, let us consider two (required) hypothetic interfaces: `SilverCustomerDBcreation` which includes `create(PI,BI):SC` and `ChildDBcreation` which includes `create(I,A,ID):C`. As `create(PI,BI):SC` is in Concepts 7, 5, 2 and 1, these concepts are used as formal attributes for interface `SilverCustomerDBcreation`. Similarly, `create(I,A,ID):C` appears in Concepts 10, 9, 5, 2, 4 and 1, giving part of the formal attributes of interface `ChildDBcreation`. This kind of encoding corresponds to existential scaling operator of the RCA approach [8]. Based on a discovered functionality of the first lattice (`create(I):SC` in Concept 5), the technique can infer a new interface, including at least this shared functionality. Here is a fundamental advantage of FCA/RCA techniques compared to the simple computing of signature comparison: new signatures emerge that provoke emergence of new interfaces which abstract the existing ones, etc. The process has to further be tuned by the direction of interfaces (provided versus required) which was not included in the above discussion for simplicity sake and reverses the specialization order on signatures.

The next generalization level concerns components. Discovering new component external specifications, in the final step of the process is of great interest for component designers because they discover abstract, more reusable components that can guide them in their development work.

6 Related Work

A service trader (yellow page mechanism) is a kind of directory that allows to index and locate components through the definition of the services they provide [9]. Existing proposals, such as CORBA Trading Object Service [10], mostly follow the ODP standard principles [11]. A component exports to the directory a service advertisement to be registered as a provider of a service. The service advertisement conforms to a service type that specifies the properties and the syntactic interfaces that a component must feature to provide this service. Service types are organized as a specialization hierarchy. Requests are sent to the directory to look for providers of some service, as defined by its type. Corresponding service advertisements can be filtered by conditions defined on property values.

Previous works do not use FCA as a base methodology. In this kind of approaches, a semi-automatic indexing methodology is proposed in [12] to help the developer identify suitable components in an existing repository. The retrieval is based on grouping names and keywords, and incremental queries that help to refine the search of a component. Within the context of identifying web services, the approaches are based on machine learning techniques, to support service classification and annotation [13, 14]. Starting from free text service documentation, services are automatically classified in classes/domains using Support Vector Machines or Ontologies. Successively, FCA is used to group text-based information related to the components to be identified. Contrary to our proposal, these approaches (some of which use FCA and some not) use an explicitly and statically constructed service type hierarchy [15]. Service type hierarchy and syntactic type hierarchy are only informally related. Moreover, only information about provided services and interfaces are considered. This content definition and organization limits the usages of this kind of component directories in dynamic, evolving and open environments.

7 Conclusions and Future Work

Applications of FCA, besides their contribution to a specific domain, participate in constructing a shared expertise about FCA-based methodologies. The contribution of this paper mainly focuses on technical aspects of encoding into formal contexts and on the definition of relevant usages of the built lattice in identifying compatible components. The main difficulty we encountered when encoding was to capture type specialization in opposite ways depending on parameter (IN / OUT) and functionality (provided / required) directions. The gain of FCA for functionality signature classification consists in proposing a classification where not only substitution and connection can be read, but also in which new, more abstract, signatures appear. These new functionality signatures enable the construction of component directories via interface and component classifications.

Perspectives include integrating complementary features of components as proposed in some component models [16, 17, 2]. Further research will also consider fully integrating such a pre-calculation in a yellow page components mechanism. Incrementally building the lattice will then be an extra advantage to manage incoming and outgoing components in a very dynamic environment.

References

1. Desnos, N., Vauttier, S., Urtado, C., Huchard, M.: Automating the building of software component architecture. In Gruhn, V., et al., eds.: Software Architecture: 3rd EWSA Workshop. Vol. 4344 of LNCS, France, Springer (2006) 228–235
2. Desnos, N., Huchard, M., Urtado, C., Vauttier, S., Tremblay, G.: Automated and unanticipated flexible component substitution. In Schmidt, H.W., et al., eds.: Proc. of 10th ACM SIGSOFT CBSE. Vol. 4608 of LNCS, USA, Springer (2007) 33–48
3. Cardelli, L.: A semantics of multiple inheritance. Vol. 173 of LNCS., Berlin, Springer-Verlag (1984) 51–64
4. Barbut, M., Monjardet, B.: Ordre et Classification. Hachette (1970)
5. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. Ordered Sets **83** (1982) 445–470
6. GaLicia: Galois lattice interactive constructor Université de Montréal. http://www.iro.umontreal.ca/∼galicia.
7. Dao, M., Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Improving generalization level in UML models iterative cross generalization in practice. In Wolff, K.E., et al., eds.: ICCS. Vol. 3127 of LNCS, USA, Springer (2004) 346–360
8. Huchard, M., Rouane Hacene, M., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Annals of Mathematics and Artificial Intelligence **49**(1-4) (2007) 39–76
9. Iribarne, L., Troya, J.M., Vallecillo, A.: A trading service for COTS components. The Computer Journal **47**(3) (2004) 342–357
10. OMG: Trading Object Service Specification v1.0. (2000)
11. International Organization for Standardization and International Telecommunication Union: ISO/IEC 13235, ITU-T X.9tr, Information Technology Open Distributed Processing ODP Trading Function. (1996)
12. Lindig, C.: Concept-based component retrieval. In Köhler, J., et al., eds.: Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs. (1995) 21–25
13. Bruno, M., Canfora, G., Penta, M.D., Scognamiglio, R.: An approach to support web service classification and annotation. In: Proc. of the EEE'05 Conference, USA, IEEE Computer Society (2005) 138–143
14. Corella, M.A., Castells, P.: Semi-automatic semantic-based web service classification. In: International Workshop on Advances in Semantics for Web Services, Springer Verlag (2006) 459–470
15. Marvie, R., Merle, P., Geib, J.M., Leblanc, S.: Type-safe trading proxies using TORBA. In: ISADS. (2001) 303–310
16. OMG: Unified modeling language: Superstructure, version 2.0 (2002) http://www.omg.org/uml.
17. Bures, T., Hnetynka, P., Plásil, F.: SOFA 2.0: Balancing advanced features in a hierarchical component model. In: SERA, IEEE Computer Society (2006) 40–48

SearchSleuth: The Conceptual Neighbourhood of an Web Query

Jon Ducrou and Peter Eklund

jrd990,peklund@uow.edu.au
School of Information Systems and Technology
University of Wollongong
Northfields Ave
Wollongong, 2522, Australia.

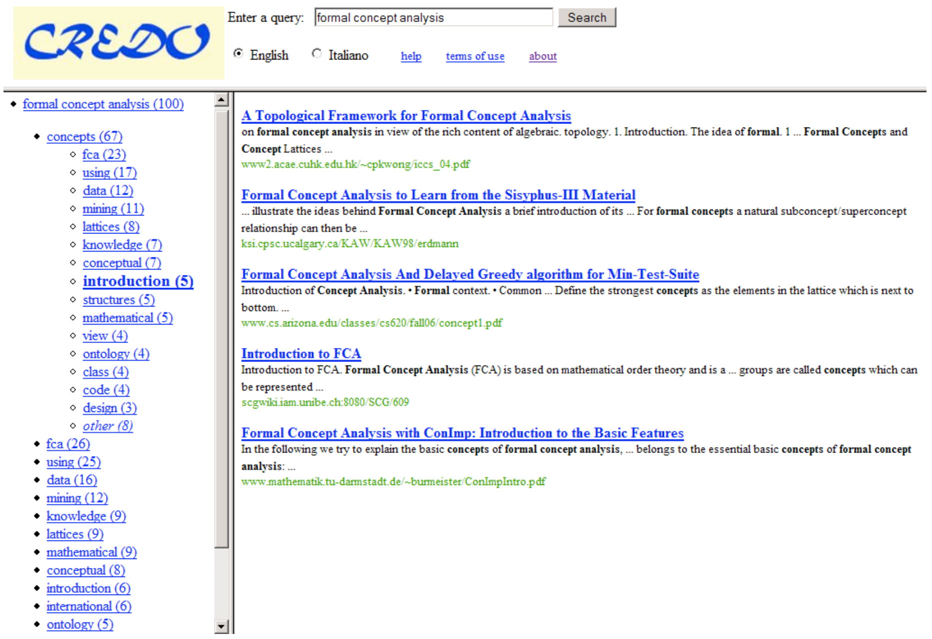
Abstract. This paper presents **SearchSleuth**, a program developed to experiment with a form of automatic local analysis that extends the standard Web search interface to include a conceptual neighbourhood focused on a formal concept derived from the query. The conceptual neighbourhood is displayed with upper neighbours representative of a generalisation operation, and lower neighbours representative of a specialisation operation. **SearchSleuth** also introduces a notion of a *categorisation* operation, where the conceptual focus can shift to a sibling concept of the search concept.

1 Introduction and Background

Existing FCA-based Web search applications focus on providing automatic local analysis of search results for query refinement and labeled object (document or document fragmets) clustering. This is usually done via the creation of a conceptual space as a concept lattice from the search results which is then displayed in various ways. Such methods fail to create a concept representing the query itself within the space – meaning the information space is representative of the results returned by the query rather than of the query itself.

SearchSleuth differs in that it creates a conceptual space as a neighbourhood of the *search concept*. This neighbourhood is comprised of generalisations (upper neighbours), specialisations (lower neighbours) and categorisation (siblings). The design therefore relies on the idea of centering the conceptual space (concept lattice) around the focal search concept. The resulting query refinement operations are more closely coupled to the search terms used in the creation of the conceptual space.

Carpineto and Romano's application, *CREDO* [1], uses an iceberg lattice to generate clusters for given search terms. Searched terms are submitted to an search engine and results returned are organized into a formal context. The context is built with the search engine results as objects and the terms found in the result summaries as attributes. The most general concepts of the concept lattice are computed and displayed as a tree. The user can then interact with the tree to view named clusters of the results. *CREDO* provides a front-end that



Enter a query: formal concept analysis Search

[✓ English] Italiano help terms of use about

- formal concept analysis (100)
  - concepts (67)
    - fca (23)
    - using (17)
    - data (12)
    - mining (11)
    - lattices (8)
    - knowledge (7)
    - conceptual (7)
    - **introduction (5)**
    - structures (5)
    - mathematical (5)
    - view (4)
    - ontology (4)
    - class (4)
    - code (4)
    - design (3)
    - *other (8)*
  - fca (26)
  - using (25)
  - data (16)
  - mining (12)
  - knowledge (9)
  - lattices (9)
  - mathematical (9)
  - conceptual (8)
  - introduction (6)
  - international (6)
  - ontology (5)

A Topological Framework for Formal Concept Analysis
on **formal concept analysis** in view of the rich content of algebraic. topology. 1. Introduction. The idea of **formal**. 1 ... **Formal Concepts** and **Concept** Lattices ...
www2.acae.cuhk.edu.hk/~cpkwong/iccs_04.pdf

Formal Concept Analysis to Learn from the Sisyphus-III Material
... illustrate the ideas behind **Formal Concept Analysis** a brief introduction of its ... For **formal concepts** a natural subconcept/superconcept relationship can then be ...
ksi.cpsc.ucalgary.ca/KAW/KAW98/erdmann

Formal Concept Analysis And Delayed Greedy algorithm for Min-Test-Suite
Introduction of **Concept Analysis**. • **Formal** context. • Common ... Define the strongest **concepts** as the elements in the lattice which is next to bottom. ...
www.cs.arizona.edu/classes/cs620/fall06/concept1.pdf

Introduction to FCA
Introduction to FCA. **Formal Concept Analysis** (FCA) is based on mathematical order theory and is a ... groups are called **concepts** which can be represented ...
scgwiki.iam.unibe.ch:8080/SCG/609

Formal Concept Analysis with ConImp: Introduction to the Basic Features
In the following we try to explain the basic **concepts** of **formal concept analysis**, ... belongs to the essential basic **concepts** of **formal concept analysis**: ...
www.mathematik.tu-darmstadt.de/~burmeister/ConImpIntro.pdf

Fig. 1. Carpineto and Romano's *CREDO* Web Search Application [1]: An example of *search*. *CREDO* displays the uppermost concepts derived from keywords featured in search results.

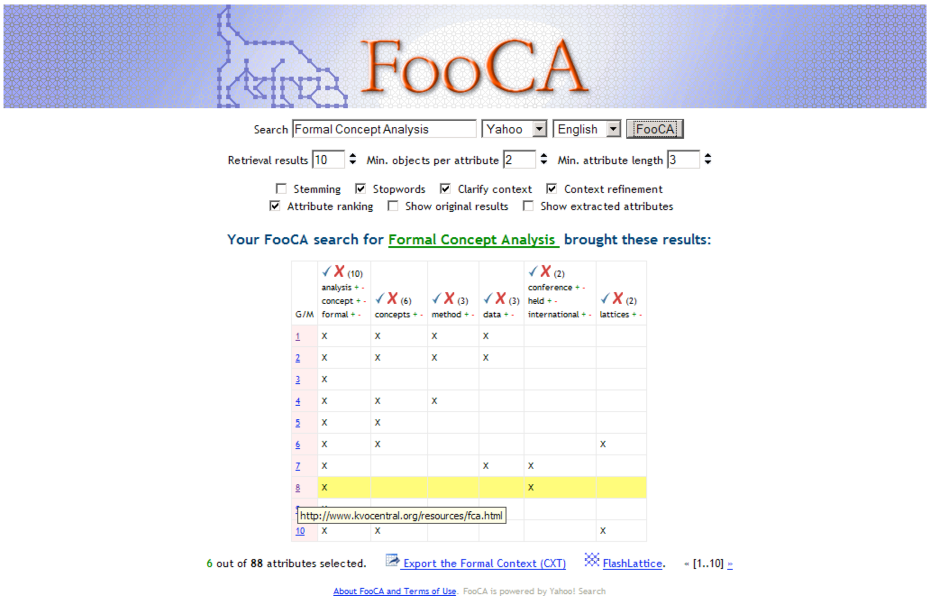
extends the collected result set, and provides a landscape over the information space which can be used to aid in search retrieval. A screenshot of *CREDO* is shown in Fig. 1.

Another FCA Web search application is Koester's FooCA [2, 3]. FooCA provides one of the richest interfaces for the construction of a formal context based on Web search results. By taking results as formal objects and terms found within the results as formal attributes, a formal context is created and presented to the user. A screenshot of FooCA is shown in Fig. 2.

2 Navigation and Conceptual Neighborhoods

Kim and Compton [4, 5], and their prototype *KANavigator*, presented a document navigation paradigm using FCA and a neighborhood display. *KANavigator* uses annotated documents that can be browsed by keyword and displayed the direct neighborhood (in particular the lower neighbours) to the user. This system emphasized the use of textual labels as opposed to visualised structure.

ImageSleuth [6] used a similar interaction approach to allow exploration of image collections. By showing upper and lower neighbors of the current formal concept and allowing the inclusion/exclusion of these formal concepts, users can refine or generalize their position in the information space by navigating the



Search Formal Concept Analysis | Yahoo | English | FooCA

Retrieval results 10 | Min. objects per attribute 2 | Min. attribute length 3

☐ Stemming ☑ Stopwords ☑ Clarify context ☑ Context refinement
☑ Attribute ranking ☐ Show original results ☐ Show extracted attributes

Your FooCA search for Formal Concept Analysis brought these results:

G/M	✓ X (10) analysis + - concept + - formal + -	✓ X (6) concepts + -	✓ X (3) method + -	✓ X (3) data + -	✓ X (2) conference + - held + - international + -	✓ X (2) lattices + -
1	x	x	x	x		
2	x	x	x	x		
3	x					
4	x	x	x			
5	x	x				
6	x	x				x
7	x			x	x	
8	x				x	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
10	x	x				x

http://www.kvocentral.org/resources/fca.html

6 out of 88 attributes selected. Export the Formal Context (CXT) FlashLattice. « [1..10] »

About FooCA and Terms of Use. FooCA is powered by Yahoo! Search

Fig. 2. Koester's FooCA Web Search Application [2, 3]: An example of *improvement*. By allowing users see and access the quality of the various terms in the current search results, successive search results increase in quality.

concept lattice. ImageSleuth was additionally aided by the use of *perspectives* – another name for conceptual scales – that could be combined to define the attribute set of the lattice which formed the information space.

ImageSleuth uses most of its interface to show thumbnails of images in the extent of the chosen concept. As a result the user never sees the line diagram of a concept lattice. Instead, the lattice structure around the current concept is represented through the list of upper and lower neighbors that allow the user to move to super- or subconcepts. For every upper neighbour (C, D) of the current concept (A, B) the user is presented with an interface that allows them to remove the set $B \setminus D$ of attributes from the current intent. Dually, for every lower neighbour (E, F) the user may include the set $F \setminus B$ of attributes which takes her to this lower neighbour. By offering the sets $B \setminus D$ and $F \setminus B$ dependencies between these attributes are shown. Moving to the next concept not having a chosen attribute in its intent, may imply the removal of a whole set of attributes. **ImageSleuth** was usability tested and results indicated that the approach used aided in navigation of image collections and it therefore gives rise to the re-use of this interaction paradigm in **SearchSleuth**.

A formal concept A is said to be the upper neighbour (or cover) of a formal concept B iff we have $A > B$, and there is no intermediate formal concept C with $A > C > B$. A formal concept A is said to be the lower neighbour of (or covered by) a formal concept B iff we have $A < B$, and there is no formal concept C with $A < C < B$. Upper and lower neighbour of a formal concept C are written as $UN(C)$ and $LN(C)$ respectively in this paper.

3 The Domain of SearchSleuth

Web search is a difficult problem given the Internet is immeasurably large and constantly being changed and altered, both in content and structure. Traditionally, search is done by entering one or more keywords into a search engine, then reviewing the results the appropriate Web site is found. This process requires a good understanding of the link between the search terms and their results.

Most search engines return a ranked list of results and these usually take the form of a document fragment or snippet that includes the URL of the result, its title, a short summary of the document and various details such as date last accessed. In CREDO and FooCA, it is these text-based snippets of the search results that provide the material to create a concept lattice that forms the basis for a conceptual information space to navigate the search results. One problem with the transition from Web search results to FCA is that the search result ranking information is lost. All results are treated equally, this issue is usually addressed by reproducing the rank ordering on any result set that is realized but this can be costly to performance.

Another problem, particularly with Web search, is that page ranking methods use techniques such as link structure analysis, page popularity and referring pages to condition page proximity to a search query. As such, that it cannot be assumed that all results of a multiple term query will contain all search terms from the original query. Even a single search term may yield search results that do not contain the search term. This may seem counter intuitive, but if there are enough Web pages linked to the result page that *does* contain that search term, that page's rank may be inflated high enough to feature in the result set.

Like *CREDO* [1] and FooCA [2, 3], **SearchSleuth** uses the 'result has term' representation for building a formal context. This means each result from the search is considered an object, and all terms contained in the results title and summary are considered attributes. In the case of **SearchSleuth** (and optionally in FooCA) all words in the result are stop-word filtered and stemmed to their lexical root. This reduces the size and complexity of the formal content by the reduction of terms with common lexical roots; displaying 'car' and 'cars' as a single term. By extension this reduces the size of the concept lattice formed from the formal context.

CREDO does not include the original query terms when creating the context. This omission is made because in most cases it would be expected that all results contain the original query terms. As Web search does not behave exclusively as a boolean search for keywords in pages, this assumption does not always hold and the original query terms may be absent from the context as previously discussed. *CREDO*, unlike **SearchSleuth**, displays two levels of the lattice as a tree with users initially placed at the top-most concept. The display is initially restricted to a single level, by with user interaction a single concept can be expanded at a time. This reduces clutter and also user confusion; users are not displayed with multiple tree branches at the same level with the same label.

The FooCA application builds and displays the entire formal context derived from the query terms returned from the search engine. The formal context is built without the originating query terms and the interface provides numerous controls over the information space created by the concept lattice. The user is shown the entire formal content in one cross-table. Cross-tables are less than perfect for human interpretation but the display does permit great detail. By viewing the entire formal content, the user is never positioned within the induced information space, and thus is without a perspective that is defined by the query.

SearchSleuth attempts to immerse the user in the information landscape, with a perspective centered on the query used to create it. The design logic is that this should give more insight into the meaning of the query with respect to other terms that appear in the result set.

Also, **SearchSleuth** is the only FCA-based Web search that uses multiple searches per query. This is done to expand the information space. The ancillary searches each yield half the results of the main query search. This expands the bounds of the information space with more general queries and provides for better clustering of facts as is explained in the next section.

4 Design Approach of SearchSleuth

SearchSleuth follows from the usability testing of **ImageSleuth** and employs the same conceptual neighbourhood paradigm for display purposes. Unlike **ImageSleuth**, **SearchSleuth**'s context is not static, so the space is rebuilt with each navigation step. This is because computing the entire domain, the Web, as a conceptual neighborhood would be computationally prohibitive.

The formal context for **SearchSleuth** is created on demand for each query; this suits the dynamic nature of the Web. The formal objects are induced from the top X results from the query, and the formal attributes are the terms contained in the title and summary of each result (like *CREDO* and *Fooca*). Terms are extracted from the title and summary after stemming and stop-word filtering has been performed. As mentioned, stemming reduces words to their lexical root (e.g. `jump`, `jumping` and `jumps` are all reduced to `jump`). Stop-word filtering removes words without individual semantic value, for example `a`, `the` and `another`. Removing these words reduces the complexity of the context without any noticeable reduction in semantic quality. For presentation purposes the stemming is reversed, this way terms rendered unintelligible by stemming are reformed in a meaningful manner.

The formal context is then reduced by removing attributes with low support. Every attribute that has less than 5% of the objects in the incidence relation is removed. This greatly decreases the computational overhead involved in most FCA algorithms. The reduction rarely effects the computed conceptual neighborhood as the terms removed are scarce within the information space.

Once the formal context is constructed, the search concept is created. This is done by taking the original query terms as attributes and deriving the formal concept. The upper neighbours of this formal concept are then derived and used to expand the formal context. This is done by querying the search engine with the attributes of each upper neighbor and inserting the results into the context. Results for these ancillary searches are limited to fewer search results.

This process of building the formal context increases the number of terms in the formal context based on a single level of generalisation. This makes the induced information space larger and richer.

Once the formal context is expanded, the search concept is recomputed as it may have been invalidated by this process. The upper and lower neighbours are computed next, then the sibling concepts. The explanation of neighbor is found in Section 2. Sibling concept are then calculated by finding all of the lower neighbours of upper neighbors which are upper neighbours of lower neighbors. Put another way, siblings are the removal of an attribute (or attributes) that defines an upper neighbor, and the inclusion of an attribute (or attributes) that defines a lower neighbor.

Consider the set of concepts X, $UN(X)$ is defined as the union of all upper neighbours of the concepts in X.

$$UN(X) := \bigcup\{UN(C) \mid C \in X\}$$

Dually, consider the set of concepts X, $LN(X)$ is defined as the union of all lower neighbours of the concepts in X.

$$LN(X) := \bigcup\{LN(C) \mid C \in X\}$$

For a lattice with concept, C, the set of concepts, S is the siblings of C, defined:

$$S := [LN(UN(C)) \cap UN(LN(C))] \backslash \{C\}$$

An example is shown in Fig. 3; concepts with a grey backing are siblings of the concept marked with a C.

Using the same labeling scheme as **ImageSleuth** for upper and lower neighbors and using the full intent as labels of sibling concepts, a display is rendered for the user. The primary feature of the display is the text entry box in which the query is entered. It is considered representative of the derived search concept, and thus is centered in the display. This is shown is Fig. 4.

Upper neighbors are shown above this text entry box, displayed as text labels. The labels are the attributes which would be removed to navigate to that upper neighbor. These labels are preceded by a minus symbol (-) – the standard exclusion operator in most search engine interfaces – to reinforce the notion of *removal*: shown in Fig. 4.

Lower neighbors are similarly displayed, but placed below the text entry box. These labels are the attributes which would be added to navigate to that lower

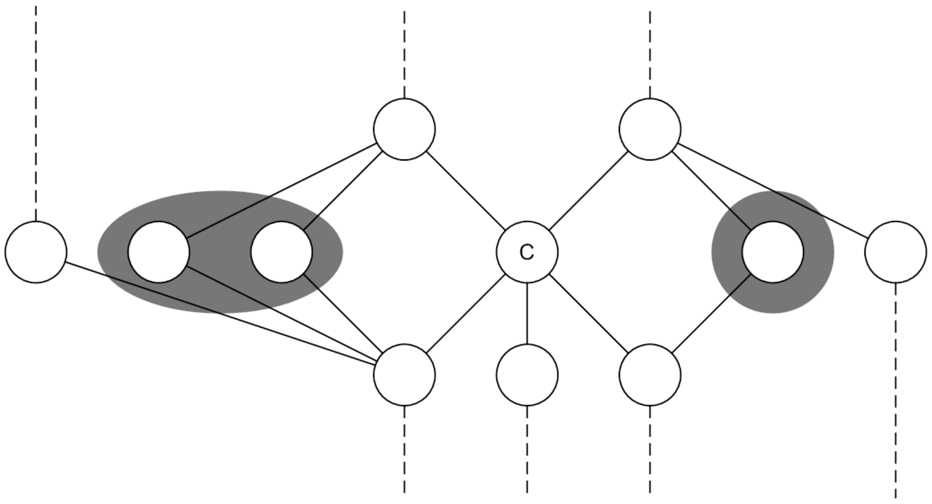


Fig. 3. Diagram demonstrating the *sibling* concepts of the concept labelled with a C.

neighbour. Like upper neighbor labels, these labels are preceded by a symbol to reinforce the labels meaning, namely the plus symbol (+) and the notion of *include*: shown in Fig. 4.

The display order of the upper and lower neighbors is determined by extent size, the concepts with larger extents displayed first (left-most). Extent is representative of the importance or prominence of a term within the current information space. This means the inclusion of the first upper or lower neighbor shown will have the least effect on the extent size current concept, if the transition were to be made on a static formal context. Extent is also used to aid in the coloring of the labels background. The larger the extent on a lower neighbor, the deeper the blue behind that concepts label. Upper neighbors are displayed with the same principle but with red.

One method for dealing with the return of empty-extents from term-based searching is to provide users with a list of the terms entered so that they can incrementally remove terms to unconstrain the search. Another method is to apply a vector space model of MPEG-7 images [8] and then apply similarity measures for multi-dimensional feature spaces. **ImageSleuth** explores a different approach by using variations on defined distance [9] and similarity [10] metrics in the FCA literature in order to find relevant concepts.

Siblings are shown to the right of the text entry box, see shown in Fig. 4. The complete intent of these concepts is displayed within square brackets preceded by a tilde symbol (`~[...]`). This helps groups the concept intents and aid distinguishing between the concepts. Unlike upper and lower neighbours, which are sorted by extent size, siblings are ordered by *similarity* to the search concept. The similarity metric is based on work by Lengnink[9] and adapted for practical application to information retrieval and browsing in **ImageSleuth**. Similarity so defined uses the size of the common objects and attributes of the concepts. For two concepts (A, B) and (C, D):

$$\mathit{similarity}((A,B),(C,D)) := \frac{1}{2}\left(\frac{|A \cap C|}{|A \cup C|} + \frac{|B \cap D|}{|B \cup D|}\right).$$

The similarity metric is used to order the sibling concepts, while highlighting remains based on the extent size for each concept. Coloring on sibling labels is based on grey shades, the darker the larger the concept's extent.

The conceptual neighborhood representation is followed by the results of the search shown in the lower half of the screen shown in Fig. 4.

By clicking any of the possible concept labels, the query is set to the intent of the selected concept and the query process is restarted. This is an important restructuring step as a change in the query in this way changes the result set, and in order for the information to be valid it needs to be recomputed.

An example of the interface is show in Fig. 4. The search concept is based on the three query terms `formal concept analysis`. It shows a single upper neighbor labeled with `-analysis` (which represented the concept with the intent (`formal, concept`)) which interestingly shows an implication that `formal` and `concept` are implied by `analysis`. The first of the lower neighbors is the acronym `fca`. This is followed by terms such as `lattice`, `mathematics` and `theory`. These terms are good examples of specialisation from the concept of Formal Concept Analysis and provide some intuitive validation of the approach.

This neighborhood is based on 115 formal objects. The initial number of formal attributes for this example was 623, after reducing the formal context this was decreased to 40. Thus, stemming and stop-word removal offer a tremendous reduction in context complexity, and therefore computation time.

Performance of the prototype shows that the vast majority of the time taken to display the results of a query is spent transferring search results from the search server. For the page shown in Fig. 4, computation of formal concepts took a total of 422ms, while transfer between Yahoo servers and the **SearchSleuth** host, comprising of two search requests, took 4062ms.

A more detailed example is shown in the next section.

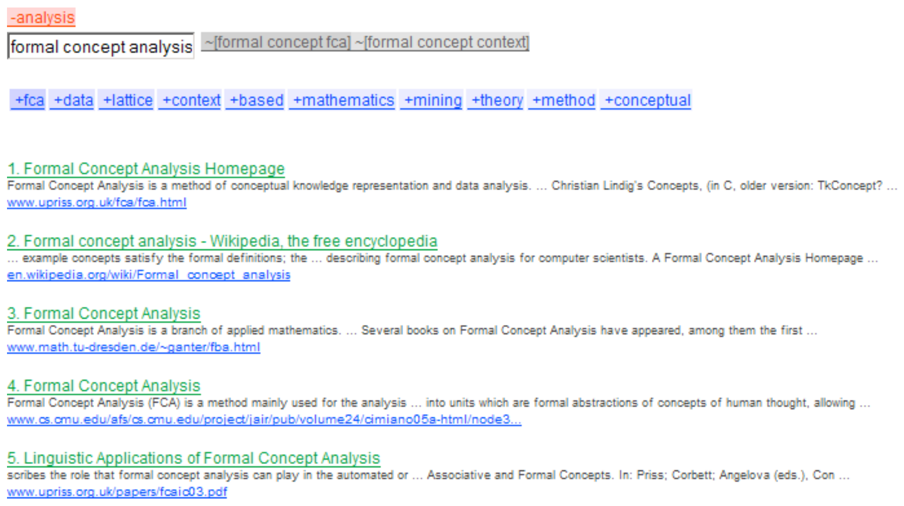
4.1 An Example Interaction for SearchSleuth

The following exemplifies a possible navigation of a dynamic information space centered with **SearchSleuth**[1], initially, on the search term '`tiger`'[2].

In Fig. 5, it can be seen that the space has no generalisations or categorisations. This is because the search concept is the top-most concept of the lattice, and therefore *all* objects have the attribute `tiger`. Examples of what may be expected as specialisations of this search concept are:

[1] the reader is encouraged to follow the example by testing **SearchSleuth** which can be accessed from http://www.kvocentral.org/software/searchsleuth.html

[2] Searching for big cat codenames for Mac OS X versions is an unofficial, but well established, standard for FCA Web-search examples.


-analysis
formal concept analysis
~[formal concept fca] ~[formal concept context]
+fca +data +lattice +context +based +mathematics +mining +theory +method +conceptual

1. Formal Concept Analysis Homepage
Formal Concept Analysis is a method of conceptual knowledge representation and data analysis. ... Christian Lindig's Concepts, (in C, older version: TkConcept? ...
www.upriss.org.uk/fca/fca.html

2. Formal concept analysis - Wikipedia, the free encyclopedia
... example concepts satisfy the formal definitions; the ... describing formal concept analysis for computer scientists. A Formal Concept Analysis Homepage ...
en.wikipedia.org/wiki/Formal_concept_analysis

3. Formal Concept Analysis
Formal Concept Analysis is a branch of applied mathematics. ... Several books on Formal Concept Analysis have appeared, among them the first ...
www.math.tu-dresden.de/~ganter/fba.html

4. Formal Concept Analysis
Formal Concept Analysis (FCA) is a method mainly used for the analysis ... into units which are formal abstractions of concepts of human thought, allowing ...
www.cs.cmu.edu/afs/cs.cmu.edu/project/jair/pub/volume24/cimiano05a-html/node3...

5. Linguistic Applications of Formal Concept Analysis
scribes the role that formal concept analysis can play in the automated or ... Associative and Formal Concepts. In: Priss; Corbett; Angelova (eds.), Con ...
www.upriss.org.uk/papers/fcaic03.pdf

Fig. 4. SearchSleuth display, including top results, after a search for '`formal concept analysis`'.

Big Cat	Adding `+cat` would probably focus on the Tiger species.
Mac OS	Adding `+os` would probably focus on the operating system (OS) used by Mac computers.
Result Facets	`+information`, `feature`, `+photo` and `+facts` are types of results, and can guide the user.

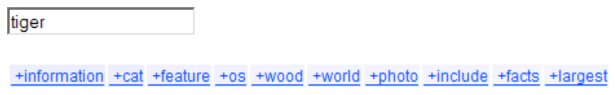

tiger
+information +cat +feature +os +wood +world +photo +include +facts +largest

Fig. 5. SearchSleuth display after a search for '`tiger`'.

Fig. 6 shows the addition of `information` via specialisation. It can be seen that this has changed the information space surrounding the query with more generic terms, such as `+pictures` and `site`. Also information about the animal tiger is made available, such as `endangered`, `conserving` and `white`. Cateogisations in this space focus on combining `tiger` with elements found by combining `tiger` with `information`, such as `tiger species` and `tiger wild`.

Fig. 7 shows the addition of `os` via specialisation from Fig. 5. This specialisation has created an information space focusing on the Apple Mac operating system, in particular version 10.4 which is also known as 'tiger'. The first two specialisations are representative of the company which makes the OS. The next two are in reference to the version of the OS. Categorizations in this space tend toward the removal of the term `tiger` in place of the focus on `os`.

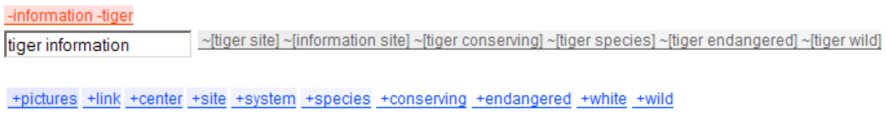

-information -tiger
tiger information
~[tiger site] ~[information site] ~[tiger conserving] ~[tiger species] ~[tiger endangered] ~[tiger wild]
+pictures +link +center +site +system +species +conserving +endangered +white +wild

Fig. 6. SearchSleuth display after clicking the '`+information`' link in Fig. 5.

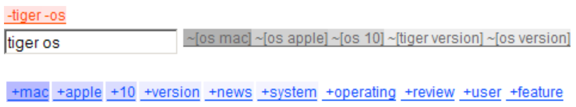

-tiger -os
tiger os
~[os mac] ~[os apple] ~[os 10] ~[tiger version] ~[os version]
+mac +apple +10 +version +news +system +operating +review +user +feature

Fig. 7. SearchSleuth display after clicking the '`+os`' link in Fig. 5.

Specialising on `cat` from Fig. 5 gives the conceptual neighborhood shown in Fig. 8.

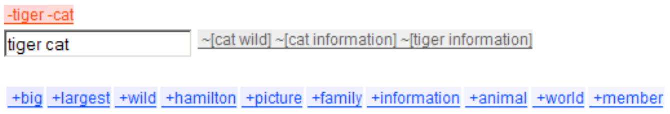

-tiger -cat
tiger cat
~[cat wild] ~[cat information] ~[tiger information]
+big +largest +wild +hamilton +picture +family +information +animal +world +member

Fig. 8. SearchSleuth display after clicking the '`+cat`' link in Fig. 5.

Specialising on `feature` from Fig. 5 reveals an interesting categorisation shown in Fig. 9. Namely a categorization concept with the label `~[tiger woods]` which most likely refers to the golfer. This reveals another possible meaning for the term `tiger`.

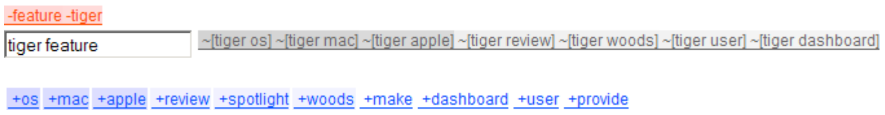

-feature -tiger
tiger feature
~[tiger os] ~[tiger mac] ~[tiger apple] ~[tiger review] ~[tiger woods] ~[tiger user] ~[tiger dashboard]
+os +mac +apple +review +spotlight +woods +make +dashboard +user +provide

Fig. 9. SearchSleuth display after clicking the '`+feature`' link in Fig. 5.

The `tiger woods` search has a definite golf focus thus confirming the navigation intuitions.

5 Conclusion

SearchSleuth, extends current FCA Web search engines by positioning the user within the information space induced by the search formal contest, rather than placing the user arbitrarily within the space or presenting the entire space. This allows generalization and categorization operations to be performed against the

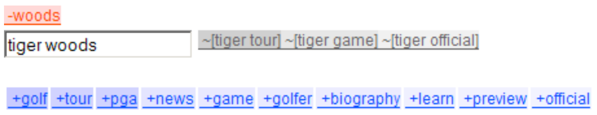

-woods
tiger woods
~[tiger tour] ~[tiger game] ~[tiger official]
+golf +tour +pga +news +game +golfer +biography +learn +preview +official

Fig. 10. SearchSleuth display after clicking the '∼`[tiger woods]`' link in Fig. 9.

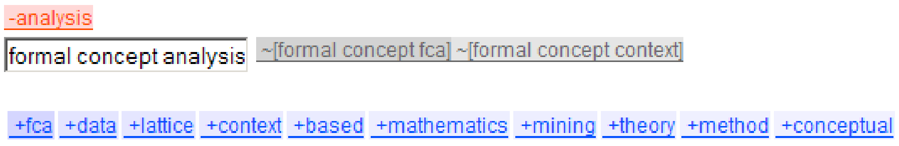

-analysis
formal concept analysis
~[formal concept fca] ~[formal concept context]
+fca +data +lattice +context +based +mathematics +mining +theory +method +conceptual

Fig. 11. SearchSleuth display after a search for '`formal concept analysis`'.

current query concept induced by the search terms. **SearchSleuth** overcomes a number of practical difficulties in the use of FCA for Web Search, namely a practical approach to the construction of a sparse formal context and the categorization operation, where the conceptual focus can shift to a sibling concept of the induced search concept.

References

1. Carpineto, C., Romano, G.: Exploiting the potential of concept lattices for information retrieval with credo. J. UCS **10**(8) (2004) 985–1013
2. Koester, B.: Fooca - web information retrieval with formal concept analysis. Diploma, Technische Universität Dresden (2006)
3. Koester, B.: Conceptual knowledge processing with google. In: Contributions to ICFCA. (2005) 178–183
4. Kim, M., Compton, P.: Incremental development of domain-specific document retrieval systems. In: 1st International Conference on Knowledge Capture. (2001)
5. Kim, M., Compton, P.: Formal concept analysis for domain-specific document retrieval systems. In: Lecture Notes in Computer Science. Volume 2256. (2001) 237
6. Ducrou, J., Vormbrock, B., Eklund, P.: FCA-based Browsing and Searching of a Collection of Images. In: Proceedings of 14th International Conference on Conceptual Structures. LNAI4068, Springer (2006) 203–214
7. Cigarrán, J.M., Gonzalo, J., Peñas, A., Verdejo, F.: Browsing search results via formal concept analysis: Automatic selection of attributes. In: Concept Lattices. Volume 2961/2004., Springer (2004) 74–87
8. Park, K.W., Lee, D.H.: Full-automatic high-level concept extraction from images using ontologies and semantic inference rules. In: The Semantic Web - ASWC 2006. LNAI4185, Springer (2006) 307–321
9. Lengnink, K.: Ähnlichkeit als Distanz in Begriffsverbänden. In G Stumme, R.W., ed.: Begriffliche Wissensverarbeitung: Methoden und Anwendungen, Springer (2001) 57–71
10. Saquer, J., Deogun, J.S.: Concept aproximations based on rough sets and similarity measures. In: Int. J. Appl. Math. Comput. Sci. Volume 11. (2001) 655 – 674

Representing Numeric Values in Concept Lattices

John L. Pfaltz

Dept. of Computer Science, Univ. of Virginia
Charlottesville, VA 22904-4740
jlp@virginia.edu

Abstract. Formal Concept Analysis is based on the occurrence of symbolic attributes in individual objects, or observations. But, when the attribute is numeric, treatment has been awkward. In this paper, we show how one can derive logical implications in which the atoms can be not only boolean symbolic attributes, but also ordinal inequalities, such as $x \leq 9$. This extension to ordinal values is new. It employs the fact that orderings are antimatroid closure spaces.

1 Extending Formal Concept Analysis

Formal Concept Analysis (FCA), which was initially developed by Rudolf Wille and Bernhard Ganter [3], provides a superb way of describing "concepts", that is closed sets of attributes, or properties, within a context of occurrences, or objects. One can regard the concept as a closed set of objects with common attributes. Frequently clusters of these concepts, together with their structure, stand out with vivid clarity. However, two unresolved problems are often encountered. First, when concept lattices become large, it is hard to discern or describe significant clusters of related concepts. Gregor Snelting used formal concept analysis to analyze legacy code [6, 14].[1] Snelting's goal was to reconstruct the overall system structure by determining which variables (attributes or columns) were accessed by which modules (objects or rows). It was hoped that the concept structure would become visually apparent. Unfortunately, the resulting concept lattice shown on page 356 of [6] is little more than a black blob. Visual interpretation of closure concepts does not seem to scale well.

Second, when the attribute values are numeric, as is often the case with technical data, formal concept analysis becomes difficult. It is easy to comprehend the set of all objects whose color is "red"; But what precisely constitutes the set of objects whose weight is "9.5", or "near 9.5", or "less than 9.5" or otherwise similar to "9.5". Fuzzy set theory [5, 18] might be appropriate here, but we prefer a deterministic approach.

To cope with the first issue, our approach has been to work with the logical implications that can be deduced from adjacent closed concepts in the lattice, rather than to seek clusters of such concepts. So long as all the attributes are boolean this seems to work rather well. We have applied formal concept analysis to relations (contexts) of sizes $8,124 \times 85$ and $1,272 \times 144$. The resulting closed set lattices were large, 104,104 and 1,804 nodes respectively, but nevertheless we were able to extract useful logical implications. In Section 3 we briefly review some of this older work.

In Section 4 we will develop our new approach to representing numeric predicates. Very briefly, it will consist of viewing a concept, not as being a closed set of attributes associated with a set of objects, but rather regarding it as a closed set of "closed sets" associated with the objects.

[1] Siff and Reps [13] published shortly after.

A crucial contribution of FCA has been to establish the role and importance of closure operators and closed sets in mathematical thinking. In Section 2 we review those closure concepts that we will need later.

2 Closure Operators and Closure Spaces

The notion of "closure" plays a major role in our representation of the real world. In particular we will be concerned with closed sets of objects, closed sets of predicates and closed sets of numbers.

2.1 Closure Concepts

By a "closure system" over a "universe" $\mathbf{U}$, we mean a collection $\mathcal{C}$ of sets $X, Y, \ldots Z \subseteq \mathbf{U}$, including $\mathbf{U}$, satisfying the property that if $X, Y \in \mathcal{C}$ then $X \cap Y \in \mathcal{C}$. The sets of $\mathcal{C}$ are said to be the closed sets of $\mathbf{U}$. As an alternative to this "intersection" characterization, one can define a closure operator $\varphi : 2^{\mathbf{U}} \rightarrow 2^{\mathbf{U}}$ satisfying the following 3 axioms for all X, Y, Z:

$X \subseteq X.\varphi$,

$X \subseteq Y$ implies $X.\varphi \subseteq Y.\varphi$

$X.\varphi.\varphi = X.\varphi$.

(For technical reasons we prefer to use suffix operator notation, so read $X.\varphi$ as "X closure".) Readily, a set X is closed in $\mathcal{C}$, if $X.\varphi = X$. The equivalence of these two alternative definitions is well known [8], and we will use both in the following sections.

Closure systems can satisfy many other axioms, and those that do give rise to different varieties of mathematical systems. If $(X \cup Y).\varphi = X.\varphi \cup Y.\varphi$ we say φ is a topological closure. If the system satisfies the "exchange axiom", that is if $p, q \notin X.\varphi$ but $q \in (X \cup \{p\}).\varphi$ then necessarily $p \in (X \cup \{q\}).\varphi$, the system can be viewed as a kind of linear algebra, or more generally a "matroid". The Galois closure we will be using in this section satisfies neither of these additional axioms. But later in Section 4, we will be using "antimatroid" closure operators, that is those which satisfy the "anti-exchange axiom" if $p, q \notin X.\varphi$ and $q \in (X \cup \{p\}).\varphi$ then $p \notin (X \cup \{q\}).\varphi$.

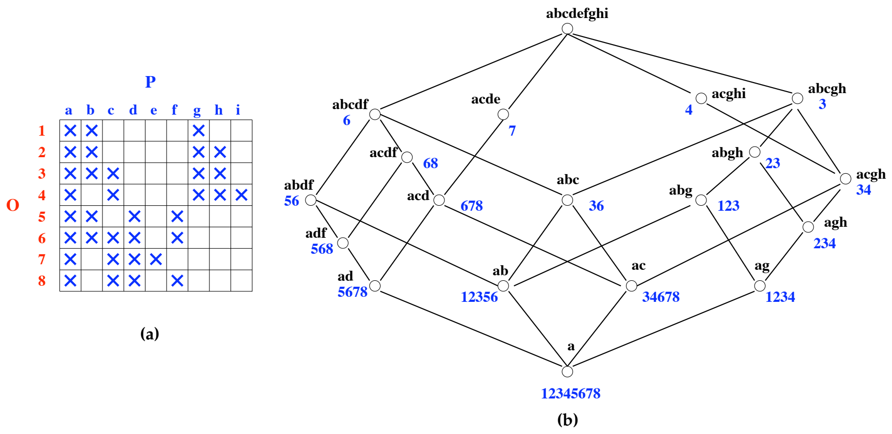
2.2 Galois Closure and Concept Lattices

The approach we will follow is similar to that was first developed by Rudolf Wille and is best presented in [3]. Formal concept analysis begins with a relation R between two sets, say a set $\mathcal{O}$ of objects and a set $\mathcal{P}$ of object predicates, or attributes. Using standard relational terminology, each object $o_j \in \mathcal{O}$ can be regarded as a row in R and each predicate $p_k \in \mathcal{P}$ is a column. Each attribute p_k is a binary, logical property, *i.e.* $p_k(o)$ is either true or false, because the object exhibits property p_k or it doesn't. A concept C_n is a pair of closed subsets $C_n = (O_n, P_n)$ where $O_n \subseteq \mathcal{O}$, $P_n \subseteq \mathcal{P}$ with the property T that for every $o_i \in O_n$, every $p_k \in P_n$, $p_k(o_i)$ is true. Each concept is assumed to be maximal, that is for the set O_n there is no larger subset $P'_n \supset P_n$ satisfying property T, and for P_n there is no larger subset $O'_n \supset O_n$ satisfying T.

The collection $\mathcal{C}$ of all concepts C_n, so defined, forms a closure system; that is, the intersection of any two concepts in $\mathcal{C}$ is a concept. Consequently, the collection $\mathcal{C}$ of concepts forms a lattice when partially ordered by containment with respect to the predicate sets P_n. [2] If we start with the relation

[2] Ganter and Wille prefer to order with respect to object set containment yielding the dual lattice, *c.f.* page 20 [3].

R of Figure 1(a) we obtain the concept lattice $\mathcal{L}$ shown as Figure 1(b).[3] Each node is labeled with a



	a	b	c	d	e	f	g	h	i
1	×	×					×		
2	×	×					×	×	
3	×	×	×				×	×	
4	×		×				×	×	×
5	×	×		×		×			
6	×	×	×	×		×			
7	×		×	×	×				
8	×		×	×		×			

Fig. 1. A relation R and its corresponding concept lattice $\mathcal{L}$

closed set of predicates and a closed set of objects satisfying those predicates, or properties. These two closed sets constitute the concept pair. For example, the combination of properties *adf* is found in rows, or objects, 5, 6 and 8.

In the case of FCA, the closure operator is a Galois closure between $\mathcal{O}$ and $\mathcal{P}$, and has been well studied [1, 2, 7] In this paper we will emphasize the closure aspect as denoted by φ, rather than the concept aspect developed in [3].

2.3 Closed Sets, Generators and Logical Implication

Let C denote a closed set. Then there is some set $A \subseteq C$ such that $A.\varphi = C$. If A is a minimal such set, w.r.t. set inclusion, we call it a generator of C denoted by $C.\gamma$, or by $A \to C$ [8]. The latter symbolism is not accidental. If φ is a Galois closure, then closed set generation and logical implication are identical [9].

Whenever the closure operator is antimatroid, there is a unique generating set for any closed set [8]. This is the property we will need in Section 4. This need not be true in the case of a Galois closure. For example, the generating sets of *abcgh* in Figure 1(b) are *bcg* and *bch*, which can be expressed logically as

$$bcg \vee bch \to abcgh.$$

We use concatenation as a short hand to express set enumeration, and when the elements are logical predicates to express conjunction. Similarly, we often suppress the dependent variable. One could expand the expression above to

$$(\forall o \in \mathcal{O})[(b(o) \wedge c(o) \wedge g(o)) \vee (b(o) \wedge c(o) \wedge h(o)) \to a(o) \wedge b(o) \wedge c(o) \wedge h(o)]$$

which seems to be unnecessarily unwieldy. Note, that the universal quantification is restricted to $\mathcal{O}$, the set of observed objects.

[3] This figure is taken directly from [3].

The issue now becomes: "given a lattice of closed concepts, such as Figure 1(b), how does one derive the generating sets?" For example, what is the generator of the closed set $acde$ in Figure 1(b)? In [4] it is shown that

Theorem 1. *Let C be a closed set with respect to φ. Let $\mathcal{F}$ denote the family of difference sets $D_k = C - C_k$ where C_k are the closed sets covered* [4] *by C in $\mathcal{L}_{\varphi}$. A set $C.\gamma \subseteq C$ is a generator of C if and only if $C.\gamma \cap D_k \neq \emptyset$ for all $D_k \in \mathcal{F}$.*

With this theorem, we can construct the generator of any closed set C as a combination of elements $e_i \in D_i = C - C_i$ where C covers C_i in $\mathcal{L}$. The node $acde$ covers only one node acd in the lattice $\mathcal{L}$, so $C - C_1 = \{e\}$. Thus, using the result above we can show that the generator $acde.\gamma = \{e\}$, or $e \to acde$. That this is true is evident from Figure 1(a). Property e is only found in observation 7, where a, c, and d are also seen. So the property e, in this case, trivially implies properties a, c and d. In the case of $abcgh$, $D_1 = \{gh\}$, $D_2 = \{c\}$ and $D_3 = \{b\}$, so bcg and bch must both be generating sets.

The intuitive understanding of Theorem 1 is simple. Let C be a closed set covering C_k. Readily we cannot have $C.\gamma \subseteq C_k$ else $C.\gamma.\varphi = C_k$. So $C.\gamma$ must have at least one element e_k that is "not in" C_k to insure $C_k \subset C$. The union of these e_k is sufficient to generate a closed set strictly larger than all the C_k. And, since C covers all of the C_k, this closed set must be C. The negative motifs "not in" and "not contained" will be repeated in Section 4. With this "covering" result, the derivation of generating sets is an inexpensive, local construction whose details can be found in [11].

2.4 Path Closures in Partially Ordered Sets

Recall that a set S is partially ordered by the relation $\leq$ if it is reflexive, weakly anti-symmetric and transitive.[5] Partially ordered sets (posets) are often represented as acyclic directed graphs, such as Figure 2, which we illustrate left to right like a number line rather than up and down in a Hasse diagram.[6]

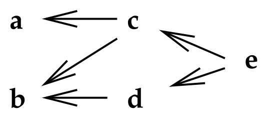


Fig. 2. A poset on 5 points.

Given any poset there are three naturally associated path closures defined
$Y.\varphi_L = \{x | x \leq y, y \in Y\}$
$Y.\varphi_R = \{z | y \leq z, y \in Y\}$
$Y.\varphi_C = \{y | y_1 \leq y \leq y_2, y_1, y_2 \in Y\}$.
These path closures have been variously called "downset" or $\downarrow$, "upset" or $\uparrow$, or "convex" closures respectively. In the case of Figure 2, these three path closures respectively yield the closed sets

[4] C covers C_k if we do not have a closed C' such that $C_k \subset C' \subset C$.
[5] It is customary to denote partial order relations with $\leq$ rather than R.
[6] Since $x \leq x, \forall x$, one could add loops at all vertices (points), but these are customarily omitted.

$\mathcal{C}_L = \mathcal{C}_{\downarrow} = \{\emptyset, a, b, ab, bd, abc, abcd, abcde\}$
$\mathcal{C}_R = \mathcal{C}_{\uparrow} = \{e, ce, de, cde, ace, acde, bcde, abcde\}$
$\mathcal{C}_C = 2^{\{abcde\}} - \{ae, be, abe, ade, bce, bde, abce, abde\}$
In the case of convex closure it is easier to enumerate the eight subsets that are not closed, *e.g.* not convex.

It is shown in [8] that these path closures on posets are antimatroid. Consequently all closed sets have unique generators. For example, consider downset closure $\downarrow$ on Figure 2. The closed sets have the indicated generating sets.

$\mathcal{C}$	∅	a	b	ab	bd	abc	abcd	abcde
$\mathcal{C}.\gamma$	∅	a	b	ab	d	c	cd	e

3 Logical Implication in Large Concept Lattices

In Section 1 we said that by interpreting a concept lattice, $\mathcal{L}$, as a set of logical implications rather than trying to analyze it visually, we could begin to cope with large lattices. We briefly review here the logical analysis of two such lattices, of 104,104 and 1,804 nodes each. Details can be found in [11, 12] and [10]. Our purpose is to reinforce the practicality of extracting generating sets (the premise(s) of the implications) using a local covering algorithm based on Theorem 1 of Section 2.3.

According to the "Audubon Society Field Guide of North American Mushrooms" mushrooms can be characterized with respect to 22 physical attributes, including "cap shape" and "cap surface". Cap shape can take the form of "bell", "conical", "flat", "knobbed", "sunken" or "convex" for 6 distinct logical attributes which we encode as $b1, c1, f1, k1, s1$ and $x1$. Similarly a cap surface can be "fibrous", "grooved", "smooth" or "scaley", which were encoded as $f2, g2, s2, y2$. By encoding the possible logical predicates for each of the 22 physical attributes, we get 85 distinct logical predicates characterizing the 8,125 different species of North American mushrooms.

The complete lattice of mushroom data has 104,104 nodes, or closed concepts. And, since each closed concept has at least one, but often many, generators we have many more logical implications — almost all of which are totally useless. But, it is not too hard to extract "useful" implications from this mass. For example, a simple AWK script that searches for the predicate $p0$ (poisonous) in the closed set (consequent) and a singleton predicate for the generator (premise) yields implications such as:

$$g2 \rightarrow p0, w3, t4, n5, f6, w7, n8$$

or, "don't eat mushrooms with grooved cap surfaces". Expanding the AWK script to retrieve similar implications, but with exactly two predicate generators yields, for example:

$$c1, y2 \rightarrow p0, n5, f6, w7, n8$$

or, "avoid those with conical, scaley caps".

An important concept in science is that of deterministic causality in which the occurrence of some event, or conjunction of events, must necessarily "cause" a consequent event. Indeed, this was the holy grail of Newtonian physics and much of 19th century science. "Causality" implies necessity, or logical implication. But, it also involves a temporal aspect. The consequent event must temporally follow all assumed antecedent events. One arena where we expect deterministic causality is software execution.

For this application we secured trace data detailing all method invocations in the transaction management module of the open source *Jboss* 1.4.2 statistical package. My colleagues, JinLin Yang and David Evans [17] had instrumented this module and generated 1,227 trace sequences consisting of 498,489 module invocations, of which 144 were distinct. Many invocations are repeatedly executed in a single trace. These yielded a $1,227 \times 144$ relation, or context, with a resultant lattice of 1,804 nodes and again many more logical implications.

These were run agains a rather simple temporal filter which ensured that all invocations in the premise (generator) preceded every invocation in the consequent (closed set) *in every trace*. This process yielded 43 likely causal dependencies of which

$17 \Rightarrow 22, 23$

$46 \Rightarrow 47, 48, 49, 60, 62$ [7]

are two representative examples. All appear to be true dependencies.

Our goal in recounting these two examples of past results is simply to establish (1) that software exists to process rather large binary relations, (2) that using Theorem 1 is a practical way of determining generator sets and (3) that considering the lattice to be a source of logical implications can have practical value. The software, which employs an incremental approach to creating the lattice, that we believe was first proposed by [15, 16], is available from the author.

4 Numeric Values

Formal Concept Analysis, as well as our own development to date, has been focused on binary predicates. Either a predicate p is true for a given object/observation, or it is not. This made it easy to reason about sets of true predicates. However, much of our understanding of the natural world is numeric. We count and we measure.

Every boolean algebra, or lattice $\mathcal{L}$, is a closure system because $x, y \in \mathcal{L}$ implies $x \wedge y \in \mathcal{L}$. In particular, any predicate $p = 0$ or 1 is a trivial boolean algebra, or closure system $\mathcal{C}_p$. Each element, 0 or 1, is its own generator. We achieve far greater expressive power if we let each predicate p_k of an observational tuple $(p_1, \ldots, p_n)$ be the generator of a closed set in a closure system $\mathcal{C}_k$, where $\mathcal{C}_k$ can be more complex than just $\{0,\ 1\}$. If each closure system $\mathcal{C}_k$ is antimatroid, as defined in Section 2.1, then every closed set has a unique generator [8]. Consequently, if the tuple of predicates $(p_1 \ldots, p_n)$ is a tuple of generators, then $(p_1 \ldots, p_n)$ denotes a unique closed set in the n-fold direct product $\mathcal{C}_1 \ldots, \mathcal{C}_n$.

It will be easier to understand the theory we are about to develop if we first consider a concrete example. Figure 3(a) has been shamelessly copied from ([3], p.44). It summarizes the ratings of 14 monuments on the *Forum Romanum* by different travel guides. Here, B = Baedecker, G = Les Guides Bleus, M = Michelin and P = Polyglott.

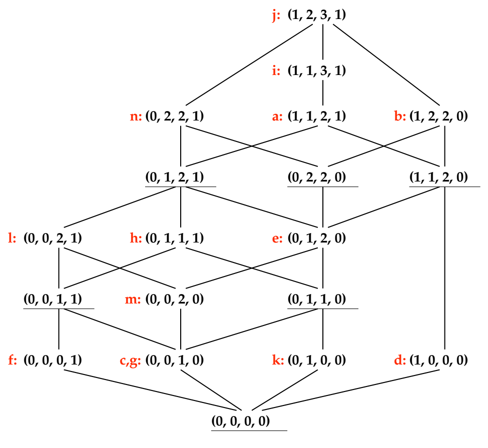
The 14 numeric 4-tuples can be partially ordered in the usual way, that is $(x_1, x_2, x_3, x_4) \leq (y_1, y_2, y_3, y_4)$ if and only if $x_k \leq y_k, k = 1, \ldots 4$. Figure 3(b), in which individual tuples have been prefixed with a letter denoting the monument giving rise to that tuple, illustrates this ordering. In order to make it a closure lattice, whenever two tuples are covered by a common tuple, their "intersection tuple" has been entered into the order, as required in any well-formed closure system. These intersection tuples have been underlined for emphasis.

Consider the tuple $n : (0, 2, 2, 1)$ which covers the tuples $(0, 1, 2, 1)$ and $(0, 2, 2, 0)$ in $\mathcal{L}$. Recall from Section 2.3 that the generators of a set are determined by the sets it covers. We claim that the generators of $n : (0, 2, 2, 1)$ are $(G > 1) \wedge (P > 0)$. Moreover, we claim that the implication embodied

[7] In a preprocessing step, each method invocation was uniquely identified by an integer, 1 through 144

		B	G	M	P
a	Arch of Septimus Severus	1	1	2	1
b	Arch of Titus	1	2	2	
c	Basilica Julia			1	
d	Basilica of Maxentius	1			
e	Phocas column		1	2	
f	Curia				1
g	House of the Vestals			1	
h	Portico of Twelve Gods		1	1	1
i	Tempel of Antonius and Fausta	1	1	3	1
j	Temple of Castor and Pollux	1	2	3	1
k	Temple of Romulus		1		
l	Temple of Saturn			2	1
m	Temple of Vespasian			2	
n	Temple of Vesta		2	2	1

(a)



(b)

Fig. 3. (a) Ratings of Roman monuments by 4 guide books, (b) Lattice of closed sets implicit in (a).

by this closed set is $[(G > 1) \wedge (P > 0)] \rightarrow [(M \geq 2) \wedge (B \geq 0)]$. First, verify that in Figure 3(a) this implication is in fact true, with j and n being confirming instances.

The principle behind this derivation, which was sketched in Section 2.3 and rigorously proven for sets in [4], is to show "what constitutes the difference between this closed set and the closed sets it covers". In our case, the only difference between $(0, 2, 2, 1)$ and $(0, 1, 2, 1)$ is that "G is not ≤ 1", or equivalently "G is > 1". Similarly, since P cannot be ≤ 0, we must have $P > 0$. The conclusion, *i.e.* those predicates with no differences, could be $B = 0$ and $M = 2$. But, what we really know is that $\neg(B < 0)$ and $\neg(M < 2)$; so $B \geq 0$ and $M \geq 2$ represent the best inferences. The adjacent tuple in Figure 3(b) is $a : (1, 1, 2, 1)$ which covers the tuples $(0, 1, 2, 1)$ and $(1, 1, 2, 0)$. Thus the same reasoning yields $[\neg(B \leq 0) \wedge \neg(P \leq 0)] \rightarrow [\neg(G < 1) \wedge \neg(M < 2)]$ or equivalently $[(B > 1) \wedge (P > 1)] \rightarrow [(G \geq 1) \wedge (M \geq 2)]$ which is verified by observations a, i and j.

Figure 4 provides a rather typical relationship between numeric data that is a bit more complex

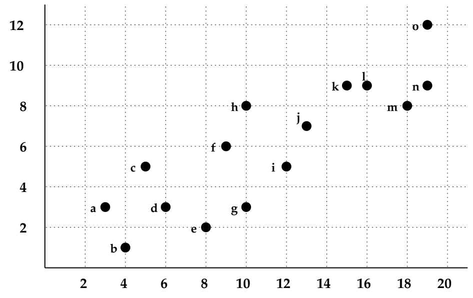


Fig. 4. A semi-random distribution of points

than Figure 3(a). We will again use downset closure, $\leq$, to order both the x and y values.

The resultant closure lattice is shown in Figure 5(b). Using the same logic that we have described

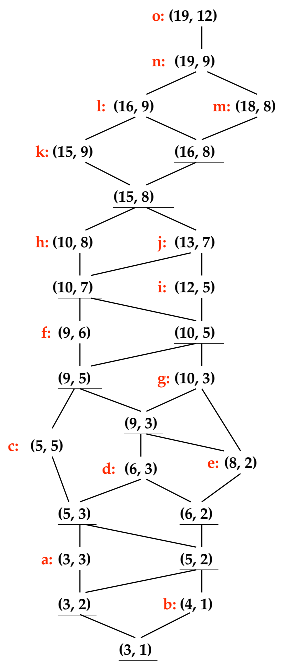


Fig. 5. The closure lattice corresponding to the points of Figure 4.

above, we can assert that $y > 3 \rightarrow x \geq 5$, based on node c; that $x > 10 \rightarrow y \geq 5$, based on node i (12,5) covering (10,5), among many others. These can be verified in Figure 5(a). We have begun to reason about the relationship between x and y.

In both the two preceding examples we used downset closure to define the numeric closure spaces. We can be more creative. For our next example we consider the same semi-random distribution of points of Figure 4, but consider 3 numeric closure spaces, specifically x with $\varphi_{\downarrow}$, y with $\varphi_{\downarrow}$ *and* y *with* $\varphi_{\uparrow}$. Thus the point i is encoded as $(12, 5, 5)$. A node $p = (x_p, y1_p, y2_p)$ is less than $q = (x_q, y1_q, y2_q)$ if $x_p \leq x_q$, $y1_p \leq y2_q$ and $y2_p \geq y2_q$. Thus $i \wedge m$ is not i as before, but rather $i \wedge m = (12, 5, 5) \wedge (18, 8, 8) = (12, 5, 8)$.

Figure 6 illustrates the resultant closed set lattice. The original points from Figure 4 are emboldened. Many of the intersection nodes are labeled with respect to these original data points. The zero element is $(3, 1, 12)$ or (min, min, MAX), while the *supremum*, or one element, is (MAX, MAX, min). We have indicated it in just this way because our software requires such a virtual supremum as its initial condition.

We observe that this lattice has a very different shape than that of Figure 5. One can see various substructures emerging within it; but nevertheless it is reaching the limit of visual comprehension.

However, we can still reason about these numeric values. For instance the node $(12, 5, 8) = i \wedge m$ covers nodes $(10, 5, 8)$ and $(12, 5, 9)$. Consequently $\neg(x \leq 10)$ and $\neg(y \geq 9)$ else $(12, 5, 8)$ could not

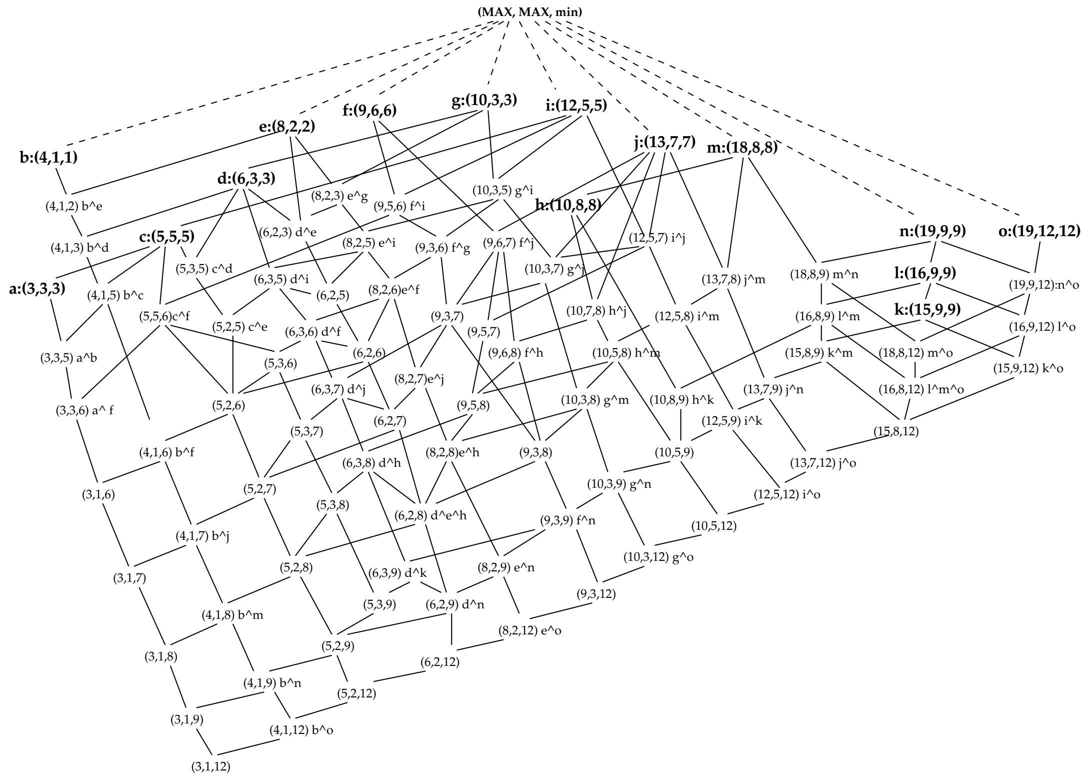


Fig. 6. The lattice of closed sets in Figure 4.

cover both nodes. Thus we know that $x > 10$ and $y < 9$ constitute the generator of $i \wedge m$, which together imply that $y \geq 5$. Inspection of Figure 5(a) shows that this describes the elements i, j, and m. The node $(16, 8, 12) = l \wedge m \wedge o$ covers only the single node $(15, 8, 12)$. If $\neg(x \leq 15)$, then because $(16, 8, 12)$ is closed we must have $y \geq 8$ and $y \leq 12)$, or more compactly, $(x > 15) \rightarrow (8 \leq y \leq 12)$. This encompasses the elements l, m, n and o.

5 Conclusions

The substitution of closed sets from an antimatroid closure space for some, or all, binary predicates in FCA is very new. We are not confident that we fully understand this process. For example, we have not yet written software that will automatically produce the lattice of closed concepts nor their generators. Figures 5 and 6 were tediously produced by hand, and even now may still contain errors. However, a method that allows us to treat numerical inequalities as boolean predicates offers considerable promise. We can imagine uncovering from legacy code rules such as:

$$\text{if not } \textit{get\_lock} \text{ and } \textit{ntries} > 10 \text{ then } \textit{abort} \qquad .$$

Undirected data mining that can yield assertions such as

$$\text{if } 5 < x < 13 \text{ then } -4 \leq y \leq 0$$

from raw numeric input could bound the error of imprecise observation.

We have applied the antimatroid path closures to totally ordered integer (or real) data sets, but they are equally defined for all partial orders. This opens the potential of using this technique to analyze the behavior of concurrent computer systems, or biological systems in which the expression of phenomena can occur along possibly multiple paths. None of this is yet reality; but the potential for greatly expanding the application of Formal Concept Analysis is here.

References

1. Gabriele Castellini. *Categorical Closure Operators*. Birkhauser, Boston, 2003.
2. Klaus Denecke, Marcel Erné, and Shelly L. Wismath. *Galois Connections and Applications*. Kluwer Academic, Dordrecht, 2004.
3. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis - Mathematical Foundations*. Springer Verlag, Heidelberg, 1999.
4. Robert E. Jamison and John L. Pfaltz. Closure Spaces that are not Uniquely Generated. *Discrete Appl Math.*, 147:69–79, Feb. 2005. also in Ordinal and Symbolic Data Analysis, OSDA 2000, Brussels, Belgium July 2000.
5. Richard Lee. Fuzzy logic and the resolution principle. *J. of the ACM*, 19(1):109–119, Jan. 1972.
6. Christian Lindig and Gregor Snelting. Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis. In *Proc. of 1997 International Conf. on Software Engineering*, pages 349–359, Boston, MA, May 1997.
7. Oystein Ore. Galois Connexions. *Trans. of AMS*, 55:493–513, 1944.
8. John L. Pfaltz. Closure Lattices. *Discrete Mathematics*, 154:217–236, 1996.
9. John L. Pfaltz. Incremental Transformations of Lattices: A key to Effective Knowledge Discovery. In A. Corradini, H. Ehrig, H-J Kreowsik, and G. Rozenberg, editors, *First Intern. Conf. on Graph Transformation*, volume Springer Verlag LNCS # 2505, pages 351–362, Barcelona, Spain, Oct. 2002.
10. John L. Pfaltz. Using Concept Lattices to Uncover Causal Dependencies in Software. In B. Ganter and L. Kwuida, editors, *Proc. Int. Conf. on Formal Concept Analysis, Springer LNAI #3874*, pages 233–247, Dresden, Feb. 2006.
11. John L. Pfaltz and Christopher M. Taylor. Closed Set Mining of Biological Data. In *BIOKDD 2002, 2nd Workshop on Data Mining in Bioinformatics*, pages 43–48, Edmonton, Alberta, July 2002.
12. John L. Pfaltz and Christopher M. Taylor. Concept Lattices as a Scientific Knowledge Discovery Technique. In *Workshop on Discrete Mathematics and Data Mining, 2nd SIAM International Conference on Data Mining*, pages 65–74, Arlington, VA, Apr. 2002.
13. Michael Siff and Thomas Reps. Identifying Modules via Concept Analysis. In *Intn'l Conf. on Software Maintenance*, pages 170–179, Bari, Italy, Oct. 1997.
14. Gregor Snelting and Frank Tip. Reengineering Class Hierarchies Using Concept Analysis. In *Proc. ACM SIGSOFT 6th International Symposium on Foundations of Software Engineering, FSE-6*, pages 99–110, Lake Buena Vista, FL, 1998.
15. Petko Valtchev, Rokia Missaoui, and Robert Godin. A Framework for Incremental Generation of Frequent Closed Itemsets. In Peter Hammer, editor, *Workshop on Discrete Mathematics & Data Mining, 2nd SIAM Conf. on Data Mining*, pages 75–86, Arlington, VA, April 2002.
16. Petko Valtchev, Rokia Missaoui, Rouane Hacene, and Robert Godin. Incremental Maintenance of Association Rule Bases. In *Proc. Workshop on Discrete Mathematic and Data Mining*, San Francisco, CA, 2003.
17. Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, and Manuvir Das. Terracotta: Mining Temporal API Rules from Imperfect Traces. In *28th Internl. Conf. on Software Engineering (ICSE 2006)*, Shanghai, China, May 2006.
18. L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

Inducing decision trees via concept lattices*

Radim Belohlavek[1,3], Bernard De Baets[2], Jan Outrata[3], Vilem Vychodil[3]

[1] Dept. Systems Science and Industrial Engineering,
T. J. Watson School of Engineering and Applied Science, SUNY Binghamton
PO Box 6000, Binghamton, New York 13902–6000, USA

[2] Dept. Appl. Math., Biometrics, and Process Control, Ghent University
Coupure links 653, B-9000 Gent, Belgium
bernard.debaets@ugent.be

[3] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
{radim.belohlavek, jan.outrata, vilem.vychodil}@upol.cz

Abstract. The paper presents a new method of decision tree induction based on formal concept analysis (FCA). The decision tree is derived using a concept lattice, i.e. a hierarchy of clusters provided by FCA. The idea behind is to look at a concept lattice as a collection of overlapping trees. The main purpose of the paper is to explore the possibility of using FCA in the problem of decision tree induction. We present our method and provide comparisons with selected methods of decision tree induction on testing datasets.

1 Introduction

Decision trees and their induction is one of the most important and thoroughly investigated methods of machine learning [4, 13, 15]. There are many existing algorithms proposed for induction of a decision tree from a collection of records described by attribute vectors. A decision tree forms a model which is then used to classify new records. In general, a decision tree is constructed in a top-down fashion, from the root node to leaves. In each node an attribute is chosen under certain criteria and this attribute is used to split the collection of records covered by the node. The nodes are split until the records have the same value of the decision attribute. The critical point of this general approach is thus the selection of the attribute upon which the records are split. The selection of the splitting attribute is the major concern of the research in the area of decision trees.

The classical methods of attribute selection, implemented in well-known algorithms ID3 and C4.5 [13, 14], are based on minimizing the entropy or information gain, i.e. the amount of information represented by the clusters of records covered by nodes created upon the selection of the attribute. In addition to that, instead of just minimizing the number of misclassified records one can minimize

* Supported by Kontakt 1–2006–33 (Bilateral Scientific Cooperation, project "Algebraic, logical and computational aspects of fuzzy relational modelling paradigms"), by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, and by institutional support, research plan MSM 6198959214.

the misclassification and test costs [9]. Completely different solutions are based on involving other methods of machine learning and data mining to the problem of selection of "splitting" attribute. For instance, in [12] the authors use adaptive techniques and computation models to aid a decision tree construction, namely adaptive finite state automata constructing a so-called adaptive decision tree. In our paper, we are going to propose an approach to decision tree induction based on formal concept analysis (FCA), which has been recently utilized in various data mining problems including machine learning via the so-called lattice-based learning techniques. For instance, in [6] authors use FCA in their IGLUE method to select only relevant symbolic (categorical) attributes and transform them to continuous numerical attributes which are better for solving a decision problem by clustering methods (k-nearest neighbor).

FCA produces two kinds of outputs from object-attribute data tables. The first one is called a concept lattice and can be seen as a hierarchically ordered collection of clusters called formal concepts. The second one consists of a non-redundant basis of particular attribute dependencies called attribute implications. A formal concept is a pair of two collections—a collection of objects, called an extent, and a collection of attributes, called an intent. This corresponds to the traditional approach to concepts provided by Port-Royal logic approach.

Formal concepts are denoted by nodes in line diagrams of concept lattices. These nodes represent objects which have common attributes. Nodes in decision trees, too, represent objects which have common attributes. However, one cannot use directly a concept lattice (without the least element) as a decision tree, just because the concept lattice is not a tree in general. See [2] and [1] for results on containment of trees in concept lattices. Moreover, FCA does not distinguish between input and decision attributes. Nevertheless, a concept lattice (without the least element) can be seen as a collection of overlapping trees. Then, a construction of a decision tree can be viewed as a selection of one of these trees. This is the approach we will be interested in in the present paper.

The reminder of the paper is organized as follows. The next section contains preliminaries from decision trees and formal concept analysis. In Section 3 we present our approach of decision tree induction based on FCA. The description of the algorithm is accompanied with an illustrative example. The results of some basic comparative experiments are summarized in Section 4. Finally, Section 5 concludes and outlines several topics of future research.

2 Preliminaries

2.1 Decision trees

A decision tree can be seen as a tree representation of a finitely-valued function over finitely-valued attributes. The function is partially described by assignment of class labels to input vectors of values of input attributes. Such an assignment is usually represented by a table with rows (records) containing values of input attributes and the corresponding class labels. The main goal is to construct a decision tree which represents a function described partially by such a table and, at the same time, provides the best classification for unseen data (i.e. generalises sufficiently).

Each inner node of a corresponding decision tree is labeled by an attribute, called a decision attribute for this node, and represents a test regarding the values of the attribute. According to the result of the test, records are split into n classes which correspond to n possible outcomes of the test. In the basic setting, the outcomes are represented by the values of the splitting attribute. Leaves of the tree cover the collection of records which all have the same function value (class label). For example, the decision trees in Fig. 1 (right) both represent the function $f : A \times B \times C \to D$ depicted in Fig. 1 (left). This way, a decision tree serves as a model approximating the function partially described by the input data.

A	B	C	$f(A, B, C)$
good	yes	false	yes
good	no	false	no
bad	no	false	no
good	no	true	yes
bad	yes	true	yes

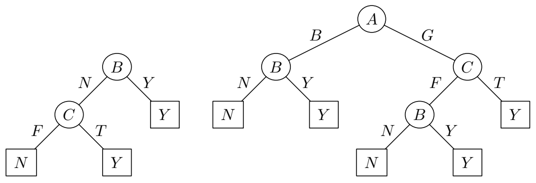


Fig. 1. Two decision trees representing example function f

A decision tree induction problem is the problem of devising a decision tree which approximates well an unknown function described partially by a relatively few records in the table. These records are usually split to two subsets called a training and testing dataset. The training dataset serves as a basis of data from which the decision tree is being induced. The testing dataset is used to evaluate the performance of the decision tree induced by the training dataset.

A vast majority of decision tree induction algorithms uses a strategy of recursive splitting of the collection of records based on selection of decision attributes. This means that the algorithms build the tree from the root to leaves, i.e. in top-down manner. The problem of local optimization is solved in every inner node. Particular algorithms differ by the method solving the optimization problem, i.e. the method of selection of the best attribute to split the records. Traditional criteria of selection of decision attributes are based on entropy, information gain [13] or statistical methods such as χ-square test [10]. The aim is to induce the smallest possible tree (in the number of nodes) which correctly decides training records. The preference of smaller trees follows directly from the Occam's Razor principle according to which the best solution from equally satisfactory ones is the simplest one.

The second problem, which is common to all machine learning methods with a teacher (methods of supervised learning), is the overfitting problem. Overfitting occurs when a model induced from training data behaves well on training data but does not behave well on testing data. A common solution to the overfitting problem used in decision trees is pruning. With pruning, some parts of the decision tree are omitted. This can either be done during the tree induction process and stop or prevent splitting nodes in some branches before reaching leaves, or after the induction of the complete tree by "post-pruning" some leaves or whole branches. The first way is accomplished by some online heuristics of classification "sufficiency" of the node. For the second way, evaluation of the ability of the tree to classify testing data is used. The simplest criterion for pruning is based on the majority of presence of one function value of records covered by the node.

2.2 Formal concept analysis

In what follows, we summarize basic notions of FCA. An object-attribute data table describing which objects have which attributes can be identified with a triplet $\langle X, Y, I\rangle$ where X is a non-empty set (of objects), Y is a non-empty set (of attributes), and $I \subseteq X \times Y$ is an (object-attribute) relation. Objects and attributes correspond to table rows and columns, respectively, and $\langle x, y\rangle \in I$ indicates that object x has attribute y (table entry corresponding to row x and column y contains $\times$; if $\langle x, y\rangle \notin I$ the table entry contains blank symbol). In the terminology of FCA, a triplet $\langle X, Y, I\rangle$ is called a formal context. For each $A \subseteq X$ and $B \subseteq Y$ denote by $A^{\uparrow}$ a subset of Y and by $B^{\downarrow}$ a subset of X defined by

$$A^{\uparrow} = \{y \in Y \mid \text{for each } x \in A:\ \langle x, y\rangle \in I\},$$
$$B^{\downarrow} = \{x \in X \mid \text{for each } y \in B:\ \langle x, y\rangle \in I\}.$$

That is, $A^{\uparrow}$ is the set of all attributes from Y shared by all objects from A (and similarly for $B^{\downarrow}$). A formal concept in $\langle X, Y, I\rangle$ is a pair $\langle A, B\rangle$ of $A \subseteq X$ and $B \subseteq Y$ satisfying $A^{\uparrow} = B$ and $B^{\downarrow} = A$. That is, a formal concept consists of a set A (so-called extent) of objects which fall under the concept and a set B (so-called intent) of attributes which fall under the concept such that A is the set of all objects sharing all attributes from B and, conversely, B is the collection of all attributes from Y shared by all objects from A. Alternatively, formal concepts can be defined as maximal rectangles of $\langle X, Y, I\rangle$ which are full of $\times$'s: For $A \subseteq X$ and $B \subseteq Y$, $\langle A, B\rangle$ is a formal concept in $\langle X, Y, I\rangle$ iff $A \times B \subseteq I$ and there is no $A' \supset A$ or $B' \supset B$ such that $A' \times B \subseteq I$ or $A \times B' \subseteq I$.

A set $\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B, B^{\downarrow} = A\}$ of all formal concepts in data $\langle X, Y, I\rangle$ can be equipped with a partial order $\leq$ (modeling the subconcept-superconcept hierarchy, e.g. *dog* $\leq$ *mammal*) defined by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \text{ iff } A_1 \subseteq A_2 \text{ (iff } B_2 \subseteq B_1). \qquad (1)$$

Note that $^{\uparrow}$ and $^{\downarrow}$ form a so-called Galois connection [5] and that $\mathcal{B}(X, Y, I)$ is in fact a set of all fixed points of $^{\uparrow}$ and $^{\downarrow}$. Under $\leq$, $\mathcal{B}(X, Y, I)$ happens to be a complete lattice, called a concept lattice of $\langle X, Y, I\rangle$, the basic structure of which is described by the so-called main theorem of concept lattices [5].

For a detailed information on formal concept analysis we refer to [3, 5] where a reader can find theoretical foundations, methods and algorithms, and applications in various areas.

3 Decision tree induction based on FCA

As mentioned above, a concept lattice without the least element can be seen as a collection of overlapping trees. The induction of a decision tree can be viewed as a selection of one of the overlapping trees. The question is: which tree do we select?

Transformation of input data Before coming to this question in detail, we need to address a particular problem concerning input data. Input data to decision tree induction contains various type of attributes, including yes/no (logical) attributes, categorical (nominal) attributes, ordinal attributes, numerical attributes, etc. On the other hand, Input data to FCA consists of yes/no attributes. Transformation of general attributes to logical attributes is known as conceptual scaling, see [5]. For the sake of simplicity, we consider input data with categorical attributes in our paper and their transformation (scaling) to logical attributes. Decision attributes (class labels) are usually categorical. Note that we need not transform the decision attributes since we do not use them for the concept lattice building step.

Name	body temp.	gives birth	fourlegged	hibernates	mammal
cat	warm	yes	yes	no	yes
bat	warm	yes	no	yes	yes
salamander	cold	no	yes	yes	no
eagle	warm	no	no	no	no
guppy	cold	yes	no	no	no

Name	bt cold	bt warm	gb no	gb yes	fl no	fl yes	hb no	hb yes	mammal
cat	0	1	0	1	0	1	1	0	yes
bat	0	1	0	1	1	0	0	1	yes
salamander	1	0	1	0	0	1	0	1	no
eagle	0	1	1	0	1	0	1	0	no
guppy	1	0	0	1	1	0	1	0	no

Fig. 2. Input data table (top) and corresponding data table for FCA (bottom)

Let us present an example used throughout the presentation of our method. Consider the data table with categorical attributes depicted in Fig. 2 (top). The data table contains sample animals described by attributes *body temperature*, *gives birth*, *fourlegged*, *hibernates* and *mammal*, with the last attribute being the decision attribute (class label). The corresponding data table for FCA with logical attributes obtained from the original ones in an obvious way is depicted in Fig. 2 (bottom).

Step 1 We can now approach the first step of our method of decision tree induction—building the concept lattice. In fact, we do not build the whole lattice. Recall that smaller (lower) concepts result by adding attributes to greater (higher) concepts and, dually, greater concepts result by adding objects to lower concepts. We can thus imagine the lower neighbor concepts as refining their parent concept. In a decision tree the nodes cover some collection of records and are split until the covered records have the same value of class label. The same applies to concepts in our approach: we need not split concepts which cover objects having the same value of class label. Thus, we need an algorithm which generates a concept lattice from the greatest concept (which covers all objects) and iteratively generates lower neighbor concepts.

For this purpose, we can conveniently use the essential ideas of Lindig's NextNeighbor algorithm [8]. NextNeighbor efficiently generates formal concepts together with their subconcept-superconcept hierarchy. Our method, which is a modification of NextNeighbor, differs from the ordinary NextNeighbor in two aspects. First, as mentioned above, we do not compute lower neighbor concepts of a concept which covers objects with the same class label. Second, unlike NextNeighbor, we do not build the ordinary concept hierarchy by means of a covering relation. Instead, we are skipping some concepts in the hierarchy. That is, a lower neighbor concept c of a given concept d generated by our method, can in fact be a concept for which there exists an intermediate concept between c and d. This is accomplished by a simple modification of NextNeighbor algorithm.

NextNeighbor The NextNeighbor algorithm builds the concept lattice by iteratively generating the neighbor concepts of a concept $\langle A, B\rangle$, either top-down the lattice by adding new attributes to concept intents or bottom-up by adding new objects to concept extents. We follow the top-down approach. The algorithm is based on the fact that a concept $\langle C, D\rangle$ is a neighbor of a given concept $\langle A, B\rangle$ if D is generated by $B \cup \{y\}$, i.e. $D = (B \cup \{y\})^{\downarrow\uparrow}$, where $y \in Y - B$ is an attribute such that for all attributes $z \in D - B$ it holds that $B \cup \{z\}$ generates the same concept $\langle C, D\rangle$ [8], i.e.

$$\begin{aligned}&(Next)Neighbors\ of\ \langle A, B\rangle = \\ &\quad \{\langle C, D\rangle \mid D = (B \cup \{y\})^{\downarrow\uparrow}, y \in Y - B\ such\ that \\ &\qquad (B \cup \{z\})^{\downarrow\uparrow} = D\ for\ all\ z \in D - B\}.\end{aligned}$$

Our modification From the monotony of the (closure) operator forming a formal concept it follows that a concept $\langle C, D\rangle$ is not a neighbor of the concept $\langle A, B\rangle$ if there exists an attribute $z \in D - B$ such that $B \cup \{z\}$ generates a concept between $\langle A, B\rangle$ and $\langle C, D\rangle$. This is what our modification consists in. Namely, we mark as (different) neighbors all concepts generated by $B \cup \{y\}$ for $y \in Y - B$, even those for which there exists a concept in between, i.e.

$$\begin{aligned}&(Our)Neighbors\ of\ \langle A, B\rangle = \\ &\quad \{\langle C, D\rangle \mid D = (B \cup \{y\})^{\downarrow\uparrow}, y \in Y - B\}.\end{aligned}$$

It is easy to see that our modification does not alter the concept lattice and the overall hierarchy of concepts, cf. NextNeighbor [8].

The reason for this modification is that we have to record as neighbors of a concept $\langle A, B\rangle$ all the concepts which are generated by the collection of attributes

B with one additional attribute. In the resulting decision tree the addition of a (logical) attribute to a concept means making a decision on the corresponding categorical attribute in the tree node corresponding to the concept. Due to lack of space we postpone a pseudocode of the algorithm of Step 1 to the full version of the paper. Part of the concept lattice built from data table in Fig. 2 (bottom), with our new neighbor relationships drawn by dashed lines, is depicted in Fig. 3.

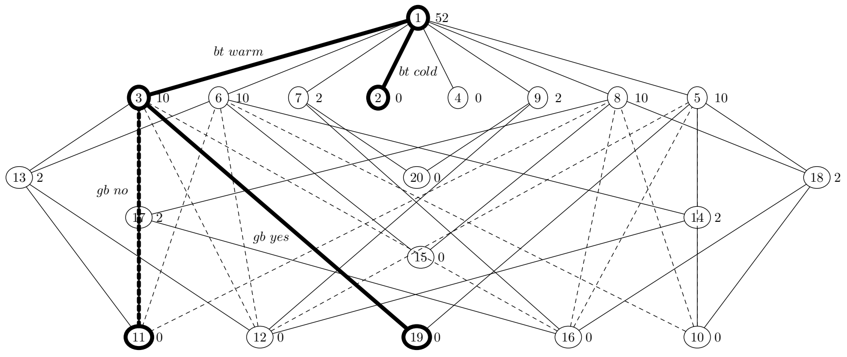


Fig. 3. Part of the concept lattice and tree of concepts (solid) of data table in Fig. 2

Step 2 The second step of our method is the selection of a tree of concepts from the part of the concept lattice built in the first step. First, we calculate for each concept $c = \langle A, B\rangle$ the number L_c of all of its lower concepts. Note that each lower concept is counted for each different attribute added to the concept c, cf. our modification of concept neighbor relation. For instance, if a concept $d = \langle C, D\rangle$ is generated from concept c by adding either attribute x or attribute y (i.e. $D = (B \cup \{x\})^{\downarrow\uparrow}$ or $D = (B \cup \{y\})^{\downarrow\uparrow}$, respectively), the concept d is counted twice and L_c is increased by two.

Next, we select a tree of concepts from the part of the concept lattice by iteratively going from the greatest concept (generated by no attributes or, equivalently, by all objects) to minimal concepts. The selection is based on the number L_c of lower concepts of the currently considered concept c (recall that L_c is not the number of lower concepts of c in common sense, cf. the computation of L_c above which is due to our modification of concept neighbor relation).

The root node of the tree is always the greatest concept. Then, for each tree node/concept c we define collections $\mathcal{N}_c^a$ of concepts, which will be candidate collections of children nodes/concepts of c in the resulting selected tree. $\mathcal{N}_c^a$ is a collection of lower neighbor concepts of c such that (a) each concept d in $\mathcal{N}_c^a$ is generated from concept c by adding a (logical) attribute transformed from the categorical attribute a (recall that logical attributes stand for values of categorical attributes) and (b) $\mathcal{N}_c^a$ contains the concept d for every logical attribute transformed from categorical attribute a. There (1) may exist, and usually exists, more than one such collection $\mathcal{N}_c^a$ of neighbor concepts of concept c, for more that one categorical attribute a, but, on the other side (2) there may exist no such collection.

(1) In this case we choose from the several collections $\mathcal{N}_c^a$ of neighbor concepts of concept c the collection containing a concept d with the minimal number L_d of its lower concepts. Furthermore, if there is more than one such neighbor concept, in different collections, we choose the collection containing the concept which covers the maximal number of objects/records. It is important to note that this point is the only non-deterministic point in our method since there still can be more than one neighbor concepts having equal minimal number of lower concepts and covering equal maximal number of objects. In that case we choose one of the collections $\mathcal{N}_c^a$ of neighbor concepts arbitrarily.

(2) This case means that in every potential collection $\mathcal{N}_c^a$ there is missing at least one neighbor concept generated by some added (logical) attribute transformed from categorical attribute a, i.e. $\mathcal{N}_c^a$ does not satisfy the condition (b). We solve this situation by substituting the missing concepts by (a copy of) the least concept $\langle Y^{\downarrow}, Y\rangle$ generated by all attributes (or, equivalently, no objects). The least concept is a common and always existing subconcept of all concepts in a concept lattice and usually covers no objects/records (but need not!).

Finally, an edge between concept c and each neighbor concept from the chosen collection $\mathcal{N}_c^a$ is created in the resulting selected tree. The edge is labeled by the added logical attribute. Again, we postpone a pseudocode of the algorithm of Step 2 to the full version of the paper.

To illustrate the previuos description, let us consider the example of a part of the concept lattice in Fig. 3. The concepts are denoted by a circled number and the number of lower concepts is written to the right of every concept. We select the tree of concepts as follows. The root node of the tree is the greatest concept 1. As children nodes of the root node are selected concepts 2 and 3 since they form a collection $\mathcal{N}_1^{body\ temp.}$ of all lower neighbor concepts generated by both added (logical) attributes *bt cold* and *bt warm*, respectively, transformed from the categorical attribute *body temp.*. Note that we could have chosen the collection $\mathcal{N}_1^{gives\ birth}$ instead of collection $\mathcal{N}_1^{body\ temp.}$, but since both concepts 2 from $\mathcal{N}_1^{body\ temp.}$ and 4 from $\mathcal{N}_1^{gives\ birth}$ have the equal minimal number L_2 and L_4 of lower concepts and both cover the equal maximal number of objects/records, we have chosen the collection $\mathcal{N}_1^{body\ temp.}$ arbitrarily, according to case (1) from the description. The edges of the selected tree are labeled by the corresponding logical attributes. Similarly, the children nodes of concept 3 will be concepts 11 and 19, and this is the end of the tree selection step since concepts 4, 11 and 19 have no lower neighbors. The resulting tree of concepts is depicted in Fig. 3 with solid lines.

Step 3 The last step (third one) of our method is the transformation of the tree of concepts into a decision tree. A decision tree has in its every node the chosen categorical attribute on which the decision is made and the edges from the node are labeled by the possible values of the attribute. The leaves are labeled by class label(s) of covered records. In the tree of concepts the logical attributes transformed from (and standing for the values of) the categorical attribute are in the labels of edges connecting concepts. Hence the transformation of the tree of concepts into a decision tree is simple: edges are relabeled to the values of the categorical attribute, inner concepts are labeled by the corresponding categorical attributes, and leaves are labelled by class label(s) of covered objects/records.

The last problem to solve is multiple different class label(s) of covered records in tree leaves. This can happen for several reasons, for example the presence of conflicting records in input data differing in class label(s) only (which can result for instance from class labelling mistakes or from selecting a subcollection of attributes from original larger data) or pruning the complete decision tree as a strategy to the overfitting problem. Common practice for dealing with multiple different target class label(s) is as simple as picking the major class label value(s) as the target classification of records covered by leave node and we adopt this solution. A special case are leave nodes represented by (a copy of) the least concept (which comes from the possibility (2) in Step 2), since the least concept usually covers no objects/records. These nodes are labelled by the class label(s) of their parent nodes.

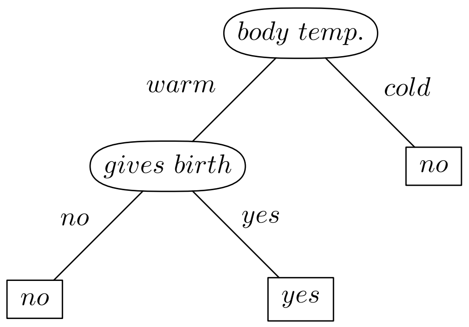


Fig. 4. The decision tree of input data in Fig. 2

The resulting decision tree of input data in Fig. 2 (top) transformed from the tree of concepts in Fig. 3 is depicted in Fig. 4.

Let us now briefly discuss the problem of overfitting. A traditional solution to overfitting problem, i.e. pruning, suggests not to include all nodes down to leaves as a part of the decision tree. One of the simplest criteria for this is picking a threshold percentage ratio of major class label value(s) in records covered by a node. Alternatively, one can use the entropy measure to decide whether the node is sufficient and need not be split. Note that several other possibilities exist. In all cases, the constraint can be applied as early as selecting the concepts to the tree, i.e. pruning can be done during decision tree induction process in our method. No pruning method is considered in this paper.

4 Comparison with other algorithms

The asymptotic time complexity of the presented algorithm is given by the (part of the) concept lattice building step since this step is the most time demanding. The concepts are computed by a modified Lindig's NextNeighbor algorithm. Since the modification does not alter the asymptotic time complexity, the overall asymptotic complexity of our method is equal to that of Lindig's NextNeighbor algorithm, namely $O(|X||Y|^2|L|)$. Here, $|X|$ is the number of input records, $|Y|$ is the number of (logical) attributes and $|L|$ is the size of the concept lattice, i.e. the number of all formal concepts.

However, for the decision tree induction problem, accuracy, i.e. the percentage of correctly and incorrectly decided records from both training and testing dataset, is more important than time complexity. We performed preliminary experiments and compared our method to reference algorithms ID3 and C4.5. We implemented our method in C language. ID3 and C4.5 were borrowed and run from the Weka[4] (Waikato Environment for Knowledge Analysis [16]), a software package which aids the development of and contains implementations of several machine learning and data mining algorithms in Java. Default Weka's parameters were used for the two algorithms and pruning was turned off where available.

Table 1. Characteristics of datasets used in experiments

Dataset	No. of attributes	No. of records	Class distribution
breast-cancer	6	138	100/38
kr-vs-kp	14	319	168/151
mushroom	10	282	187/95
vote	8	116	54/62
zoo	9	101	41/20/5/13/4/8/10

The experiments were done on selected public real-world datasets from UCI machine learning repository [7]. The selected datasets are from different areas (medicine, biology, zoology, politics and games) and all contain only categorical attributes with one class label. The datasets were cleared of records containing missing values and actually, we selected subcollections of less-valued attributes of each dataset and subcollections of records of some datasets, due to computational time of repeated executions on the same dataset. The basic characteristics of the datasets are depicted in Tab. 1. The results of averaging 10 executions of the 10-Fold Stratified Cross-validation test (which gives total of 100 executions for each algorithm over each dataset) are depicted in Tab. 2. The table shows average percentage rates of correct decisions for both training (upper item in the table cell) and testing (lower item) dataset part, for each compared algorithm and dataset. We can see that our FCA based decision tree induction method outperforms C4.5 on all datasets, with the exception of mushroom, by 2 – 4 %,

[4] Weka is a free software available at `http://www.cs.waikato.ac.nz/ml/weka/`

on both training and testing data, and gains almost identical results as ID3, again on all datasets except mushroom, on training data, but slightly outperforming ID3 on testing data by about 1 %. On mushroom dataset, which is quite sparse comparing to the other datasets, our method is little behind ID3 and C4.5 on training data, but, however, almost equal on testing data. The datasets vote and zoo are more dense than the other datasets and also contain almost no conflicting records, so it seems that the FCA based method could give better results that traditional, entropy based, methods on clear dense data. However, more experiments on additional datasets are needed to approve this conclusion.

Table 2. Percentage correct rates for datasets in Tab. 1

training % *testing %*	breast-cancer	kr-vs-kp	mushroom	vote	zoo
FCA based	88.631	84.395	96.268	97.528	98.019
	79.560	74.656	96.284	90.507	96.036
ID3	88.630	84.674	97.517	97.528	98.019
	75.945	74.503	96.602	89.280	95.036
C4.5	86.328	82.124	97.163	94.883	96.039
	79.181	72.780	96.671	86.500	92.690

Due to lack of space only the basic experiments are presented. More comparative tests on additional datasets with additional various machine learning algorithms like Naive Bayes classification or Artificial Neural Networks trained by back propagation [11], including training and testing time measuring, are postponed to the full version of the paper. However, the first preliminary experiments show that our simple FCA based method is promising in using FCA in the decision tree induction problem.

The bottleneck of the method could be performance, the total time of tree induction, but once one already has the (whole) concept lattice of input data, then the tree selection is very fast. This draws a possible usage and perspective of the method: decision making from already available concept lattices. The advantage of our method over other methods is the conceptual information hidden in tree nodes (note that they are in fact formal concepts). The attributes in concept intents are the attributes common to all objects/records covered by the concept/tree node, which might be usefull information for furher exploration, application and interpretation of the decision tree. This type of information is not (directly) available by other methods, for instance classical entropy based.

5 Conclusion and topics of future research

We have presented a simple novel method of decision tree induction by selection of the tree of concepts from a concept lattice. The criterion of choosing an attribute based on which the node of the tree is split is determined by the number of all lower concepts of the concept corresponding to the node. The approach interconnects areas of decision trees and formal concept analysis. We have also presented some comparison to classical decision tree algorithms, namely ID3 and C4.5, and have seen that our method compares quite well and surely deserves more attention. Topics for future research include:

– Explore the possibility to compute a smaller number of formal concepts from which the nodes of a decision tree is constructed. Or, the possibility to compute right the selected concepts only.
– The problems of overfitting in data and uncomplete data, i.e. data having missing values for some attributes in some records.
– Incremental updating of induced decision trees via incremental methods of building concept lattices.

References

1. Belohlavek R., De Baets B., Outrata J., Vychodil V.: Trees in Concept Lattices. In: Torra V., Narukawa Y.,Yoshida Y. (Eds.): Modeling Decisions for Artificial Intelligence: 4th International Conference, MDAI 2007, *Lecture Notes in Artificial Intelligence* **4617**, pp. 174–184, Springer-Verlag, Berlin/Heidelberg, 2007.
2. Belohlavek R., Sklenar V.: Formal concept analysis constrained by attribute-dependency formulas. In: B. Ganter and R. Godin (Eds.): ICFCA 2005, *Lecture Notes in Computer Science* **3403**, pp. 176–191, Springer-Verlag, Berlin/Heidelberg, 2005.
3. Carpineto C., Romano G.: *Concept Data Analysis. Theory and Applications.* J. Wiley, 2004.
4. Dunham M. H.: *Data Mining. Introductory and Advanced Topics.* Prentice Hall, Upper Saddle River, NJ, 2003.
5. Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.
6. Mephu Nguifo E., Njiwoua P.: IGLUE: A lattice-based constructive induction system. *Intell. Data Anal.* **5**(1), pp. 73–91, 2001.
7. Newman D. J., Hettich S., Blake C. L., Merz C. J.: *UCI Repository of machine learning databases*, [`http://www.ics.uci.edu/~mlearn/MLRepository.html`], Irvine, CA: University of California, Department of Information and Computer Science, 1998.
8. Lindig C.: Fast concept analysis. In: Stumme G.: *Working with Conceptual Structures – Contributions to ICCS 2000.* Shaker Verlag, Aachen, 2000, 152–161.
9. Ling Ch. X., Yang Q., Wang J., Zhang S.: Decision Trees with Minimal Costs. *Proc. ICML 2004.*
10. Mingers J.: Expert systems – rule induction with statistical data. *J. of the Operational Research Society.*, **38**, pp. 39–47, 1987.
11. Mitchell T.M.: *Machine Learning.* McGraw-Hill, 1997.
12. Pistori H., Neto J. J.: Decision Tree Induction using Adaptive FSA. *CLEI Electron. J.* **6**(1), 2003.
13. Quinlan J. R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
14. Quinlan J. R.: Learning decision tree classifiers. *ACM Computing Surveys*, **28**(1), 1996.
15. Tan P.-N., Steinbach M., Kumar V.: *Introduction to Data Mining.* Addison Wesley, Boston, MA, 2006.
16. Witten I. H., Frank E.: *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, San Francisco, 2005.

Policies Generalization in Reinforcement Learning using Galois Partitions Lattices

Marc Ricordeau and Michel Liquière

mricorde@wanadoo.fr, liquiere@lirmm.fr
Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier
161 rue Ada 34392 Montpellier Cedex 5 France

Abstract. The generalization of policies in reinforcement learning is a main issue, both from the theoretical model point of view and for their applicability. However, generalizing from a set of examples or searching for regularities is a problem which has already been intensively studied in machine learning. Our work uses techniques in which generalizations are constrained by a language bias, in order to regroup similar states. Such generalizations are principally based on the properties of concept lattices. To guide the possible groupings of similar environment's states, we propose a general algebraic framework, considering the generalization of policies through a set partition of the states and using a language bias as an *a priori* knowledge. We give an application as an example of our approach by proposing and experimenting a bottom-up algorithm.

1 Introduction

The reinforcement learning domain gives the promising theoretical framework of an agent learning a behavior by interactions with an environment with the following properties: (1) The agent doesn't have any *a priori* knowledge, on the environment. (2) There is no separation between a learning phase and a phase of use of the learning. (3) The agent has to manage the temporal difference learning problem, that is to say considering that the effect of an action can be delayed or that an effect can be the consequence of several successive actions.
With the formalism used to realize and demonstrate the properties of most of the algorithms as well as in practice, the approach is severely more restrictive: environments are designed with a Markovian Decision Process (MDP), the succession of interactions and perceptions are discretized and the agent is supposed to perceive its environment completely.
The constant progress of reinforcement learning since the emergence of its most famous algorithm, *Q-Learning* [1], aims at adapting the algorithms to a wider class of problems than those that can be formalized with a MDP: extension of the formalism [2] or more recently [3], environment's states or continuous actions [4], non markovian environments, partially observable environments [5].
Presently, the question of generalization of policies for reinforcement learning and their corollaries: the use of *a priori* knowledge and the search for important description features to learn the task, has become an important research fields.

Several ways are intensively studied in this frame: generalization by function approximation, algebraical model generalization [6], [7], temporal abstraction generalization [8], link with planning and ILP methods [9] or search for relevant descriptions of the environment [3].
In our work, we consider the generalization of policies and task description problem from a double point of view: the learning of a behavior by reinforcement and the problem of grouping elements which are considered similar agreeing with a description language. We propose to apply to reinforcement learning, learning methods, based on the Galois lattice algebraical structure, used in classification and more recently in data mining.
In [10], methods in the same framework have been presented but in a different way. Notably, we present here an original search space: the descriptions partitions Galois lattice instead of the classical powerset Galois lattices. Moreover, we characterize algebraically the space search, instead of building the whole lattices structures in our algorithms.
Section 2 recapitulates briefly the basis of reinforcement learning, essentially to present formalisms used in this paper. In particular, we remind the reader how to formalize the generalization of policies as a partitioning of the environment's states. Then, section 3 shows how to use Galois lattices as a generalization space for a set constrained with a description language (possibly structural). Moreover, we present the descriptions partitions Galois lattice, which is an extension of this method to partitioning a set of objects. In section 4, we see how to use this algebraical structure for policies generalization in reinforcement learning. We study the existence of solutions for the learning problem as a function of the description language. Then, in section 5, we present an algorithm based on these methods and we apply it to an academic problem. We show notably the possibility to extract relevant description elements for the task to be learned by the agent.

2 Basis of Reinforcement Learning, policies generalization

2.1 Reinforcement Learning Problem

We only give here the basis and main formalisms of reinforcement learning. A complete introduction can be found in [11]. The reinforcement learning proposes a framework where an agent learns a task in a given environment, only by receiving rewards when the task is accomplished. A reinforcement learning is classically formalized as a succession of interactions. The agent receives at the step t a **state** s_t from the environment and a numerical value of **reinforcement** $r_t \in \mathbb{R}$, called **reward**, indicating the quality of the state in which the agent is. Then, the agent acts on its environment, selecting one of the eligible **actions** a_t for the state s_t, which modifies the environment's state s_{t+1}. This process continues indefinitely or until a terminal state is reached.
The goal of the agent is to learn to select the actions which maximize the flow of the rewards $r_i, i \geq t$ to come and not only the following reward r_{t+1}. However,

the closest rewards (in term of number of interactions) are considered as more important then the far ones. Thus, the goal of the agent is to maximize the following value called **cumulated delayed reward**: $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},\ 0 < \gamma \leq 1$.

The environment is formalized with a MDP. That is to say that the state s_{t+1} depends only on the state s_t, on the selected action a_t and on a probability transition $P(s_t, a_t, s_{t+1})$. In particular, the probability doesn't depend on the previous states-actions sequence $s_0, a_0, \ldots, s_{t-1}, a_{t-1}$.
Formally, an environment is defined with a tuple $\langle S, A, \psi, P, R\rangle$; $S = \{s_1,\ s_2,\ \ldots,\ s_n\}$ is a set of states; $A = \{a_1, a_2, \ldots, a_n\}$ is a set of actions; $\psi \subseteq S \times A$ is a set of eligible couples $(state, action)$. We note $A(s)$ the set of eligible actions for the state s $((s, a_i) \in \psi)$; $P : \psi \times S \rightarrow [0, 1]$ is the set of transition probabilities from the state s to the state s' selecting the action a; $R : \psi \rightarrow \mathbb{R}$ is the reward function giving a numerical value for each eligible couple (s, a).
The function $\pi : \psi \rightarrow [0, 1]$ gives the probability for the agent to select the action a being in the state s. The function π is called the **policy** of the agent.
The function $Q^{\pi} : \psi \rightarrow \mathbb{R}$, called **quality** function, defines the expected cumulated delayed reward (expected because the process is stochastic and not deterministic) the agent would obtain following the policy π from the state s and taking the action a.
The set of all policies can have a partial order relation, based the cumulated delayed reward. A policy π_1 is greater than a π_2 if $\forall (s, a) \in \psi,\ Q^{\pi_1}(s, a) \geq Q^{\pi_2}(s, a)$. The policies allowing the agent to collect the maximum possible rewards (R_t) are called **optimal policies**. All the optimal policies share the same values for the function $Q(s, a)$ which is noted $Q^*(s, a)$. Thus, we note π^* to refer to an optimal policy.
The knowledge of $Q^*(s, a)$ allows to find greedily an optimal policy π^* by selecting in each state s, one of the actions a with a maximal expected cumulated delayed reward. We note $A^*(s)$ the set of the optimal actions for the state s.
The reinforcement learning algorithms are based on the approximation of the $Q^*(s, a)$ function, during the interactions of the agent with the environment. The hypothesis on the $Q(s, a)$ function is modified with an update rule. For example, this is the update rule for the *Q-learning* algorithm [1]:

$$Q(s, a) \leftarrow (1-\alpha)Q(s, a) + \alpha\big(r_t + \gamma \underset{a \in A(s_{t+1})}{Max} Q(s_{t+1}, a)\big),\ 0 < \alpha \leq, 0 \leq \gamma \leq 1 \quad (1)$$

Some main issues characterize the reinforcement learning. Firstly, the temporal difference learning problem, that is to say the repartition of an acquired experience (a value r_t) over the previously selected actions. Secondly, the *exploration-exploitation* problem, that is to say the balance between using the actual knowledge of $Q(s, a)$ function to maximize the flow of rewards (exploitation) and increasing its precision by selecting assumed non-optimal actions (exploration). One of the corollaries is the balance between keeping the actual theory and incorporating the newly acquired knowledge (managed with the γ parameter).
Finally, the question we propose to give elements on: the question of the generalizing of policies. It can be viewed as: "How a behavior learned in an environment

can be used in a similar environment", or "How a learned knowledge on a set of states can be generalized on a greater set of states". This problem is related to a main question of machine learning: "How to find a relevant description for a given task ?"

2.2 Generalization by partitioning the environment's states

In [7], [6], the generalization for MDPs and consequently for reinforcement learning is presented as the search for an equivalence relation between the environment's states. Two similar states can be merged if they are equivalent, i.e. if the merging doesn't change the dynamic of the environment. It's actually a generalization, because a learning on a state s is applied to the set of the states equivalent with s. However, in these articles, the equivalence relation between the states is computed directly from the MDP, that is to say from the model of the environment. In our work, we want to apply generalization methods without this information, preserving the *agent* point of view.
The equivalence relation on S, the set of the environment's states, we propose is not based on the compliance with the environment's dynamic, but on the equivalence of their optimal actions.

Definition 1. *Let $s_1, s_2 \in S$, be two environment's states. s_1 and s_2 are equivalent, which is noted $s_1 \equiv^* s_2$ if and only if $A^*(s_1) = A^*(s_2)$. The equivalence relation $\equiv^*$ implies a partitioning of the set S, noted $\mathcal{P}^*$.*

This means that all the optimal actions of s_1 are the sames (considering their label) as the optimal actions of s_2. It's important to note that we have introduced a first *a priori* knowledge since this definition implies that two actions can be equivalent according to their description.
The partition $\mathcal{P}^*$ allows to express rules such as "in the set of states S_1, we must select an action among $A^*(S_1)$ in order to act optimally". The knowledge of this set of rules, gives the optimal policies and enables us to present them in a more intelligible and general way than an classical probabilities function. Thus, the search for $\mathcal{P}^*$ becomes in our framework, the goal of the learning. The space search becomes the set partitions of S.
However, grouping the states according to their optimal actions doesn't give a direct generalization method. To do it, we need to be able to compare the states. In the next section, we show how the Galois Lattice algebraical structure can be used to generalize a set of objects with a generalization language.

3 Galois lattice as generalization space

The definitions and theorems used here about Galois lattices can be found in [12]. First, we show how this structure is used in machine learning to allow a generalization over a set of elements biased with a language. Secondly, we give an extend of Galois lattice by presenting the partitions Galois lattice.

3.1 Galois lattice and generalization languages

The Galois lattice, or more generally the lattice structure is commonly used in machine learning, directly or in an underlying way. The first well known use of such mathematical objects is certainly the Version Space [13]. More recently, the formal concept analysis theory [12] uses directly Galois lattices as structure used for generalization over a set of objects. We give here a general algebraical approach that can be found in [14].

A Galois lattice can be classically summed up as a structure defined by a double closure relation between two join semi-lattices. We note it $\mathcal{GL}(A, \leq_A, \otimes_A, B, \leq_B, \otimes_B, f, g)$, $(A, \leq_A, \otimes_A)$ and $(B, \leq_B, \otimes_B)$ being the two join semi-lattices defined respectively by a set, a partial order relation and a product operator producing the supremum of two elements and f and g being the two closure operators. In the frame of machine learning, the Galois Lattice produced from the powerset of objects and a generalization langage $(L, \leq_L, \otimes_L)$ defines the set of all the subsets authorized by a given description language and their associated description.

More formally, to use a Galois lattice as a generalization space for a set of objects E constrained with a generalization language L, we need to identify the elements of the Galois lattice definition: (1) The set A with $\mathcal{P}(E)$, the powerset of the elements to classify. (2) The partial order relation $\leq_A$ with the set inclusion $\subseteq$. (3) The product operator $\otimes_A$ with the set intersection $\cap$. (4) The set B, $\leq_B$ and $\otimes_B$ can be identified with any description language L with a partial order relation $\leq_L$ and a product operator $\otimes_L$ such that exists a description function d and an instantiation function i described below and agreeing with the Galois connection conditions. Such languages are called **generalization languages**. The relation $\leq_L$ defines the relation *more general than* or *more specific than* and $\otimes_L$ is called *generalization operator*. (5) The function f with the following description function $d : \mathcal{P}(E) \to L$, $d(P) = \bigotimes_L \{d(e), e \in E\}$. The description $d(P)$ of a subset $P \subseteq E$ is the least general generalization of the descriptions of the elements $e \in P$. Note that only the description $d(e)$, $e \in E$ must be known in order to compute the description of all subset $P \subseteq E$. (6) The function g with the following instantiation function $i : L \to \mathcal{P}(E)$: $\forall l \in L$, $i(l) = \{e \in E \mid d(e) \leq_L l\}$. The instantiation of an element of the language $l \in L$ is the set of all the elements $e \in E$ which description is more specific than l.

Finally, the set $\mathcal{C}$ of the couples $(c_a, c_b) \in C_{i \circ d}(\mathcal{P}(E)) \times C_{d \circ i}(L)$ is called the set of **concepts** of $\mathcal{GL}(\mathcal{P}, \subseteq, \cap, L, \leq_L, \otimes_L, d, i)$. Let a concept $c = (P, l)$, $P \in C_{i \circ d}(\mathcal{P}(E))$, $l \in C_{d \circ i}(L)$, P is the **extension** of c and l the **intention** of c.

We have three main properties about this structure. First, the set $C_{i \circ d}(\mathcal{P}(E))$ is the powerset of E that can be expressed with the language L. Thus, L implies a selection between all the possible generalizations of the elements of E. Roughly speaking, we can also say that the language L implies rules like "if we decide to put together the elements e_1 and e_2, then we must also regroup them with e_3". Secondly, the set $C_{d \circ i}(L)$ is the set of the descriptions of the elements of $C_{i \circ d}(\mathcal{P}(E))$. If a new, unknown element e_n occurs, we can use its description $d(e_n)$ to compare and eventually classify it in a previously found group. Finally, the set C of concepts has a lattice structure, which can be used to compare and

merge generalizations of elements of E.
This framework is very large, every generalization language L provided with $\leq_L$ and $\otimes_L$ is usable. For example, the formal concept analysis theory studies particularly the attributes-values description case with: L is the powerset $\mathcal{P}(A)$ of a set of attributes A, $\leq_L$ is the set inclusion $\subseteq$, $\otimes_L$ is the set intersection $\cap$. Notably, we can also use structural languages (see [14] for examples). Thus, the descriptions of the generalizations keep their structural properties.

3.2 Descriptions Partitions Galois lattice

We present a special case of Galois lattice: the descriptions partitions Galois lattice as generalization space for the set partitions. It is built in a similar way than the powerset Galois lattice. It will be used to constrain the possible set partitions with a generalization language describing the elements of E, with the same properties. Figure 1 presents the general scheme.

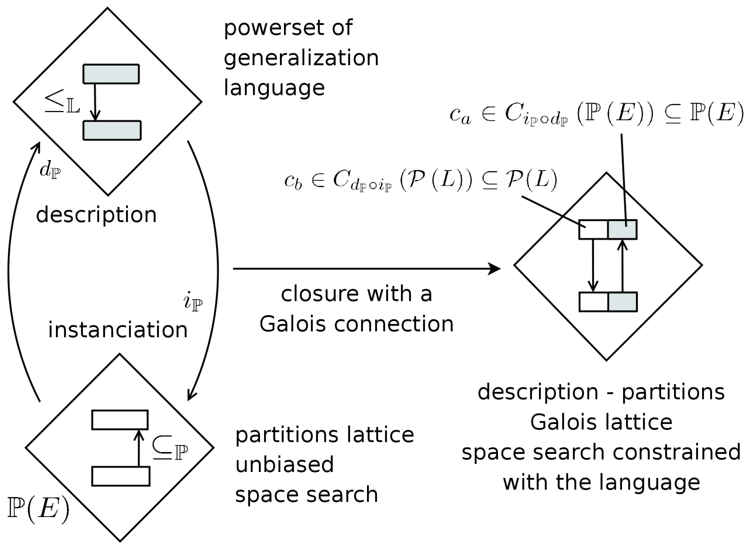

powerset of generalization language
$\leq_{\mathbb{L}}$
$d_{\mathbb{P}}$
description
$i_{\mathbb{P}}$
instanciation
closure with a Galois connection
$c_a \in C_{i_{\mathbb{P}} \circ d_{\mathbb{P}}}(\mathbb{P}(E)) \subseteq \mathbb{P}(E)$
$c_b \in C_{d_{\mathbb{P}} \circ i_{\mathbb{P}}}(\mathcal{P}(L)) \subseteq \mathcal{P}(L)$
$\subseteq_{\mathbb{P}}$
$\mathbb{P}(E)$
partitions lattice unbiased space search
description - partitions Galois lattice space search constrained with the language

Fig. 1. Scheme of partitions Galois lattice constrained with a generalization language

The main principle of the Galois lattices for partitions is to use the partitions lattice of E instead of powerset lattice and to use the powerset lattice of the generalization language L instead of L. We also have to redefine the description and instantiation functions to fit the definition. Finally, we show that the extents of the generated concepts are the partitions of E for which each element is a concept of the classical Galois lattice.
Let's define the partial order relation $\subseteq_{\mathbb{P}}$ for partitions, the product $\otimes_{\mathbb{P}}$ and the sum $\oplus_{\mathbb{P}}$ of two partitions.

Definition 2 **($\subseteq_{\mathbb{P}}, \otimes_{\mathbb{P}}, \oplus_{\mathbb{P}}$).** *Let E be a set and $\mathbb{P}(E)$ be set partitions of E. Let $\mathcal{P}_1$, $\mathcal{P}_2 \in \mathbb{P}(E)$ be two partitions of E. $\mathcal{P}_1 \subseteq_{\mathbb{P}} \mathcal{P}_2 \Leftrightarrow \forall P_1 \in \mathcal{P}_1, \exists P_2 \in \mathcal{P}_2$ such that $P_1 \subseteq P_2$.*
*Let $R_{\otimes}$ and $R_{\oplus}$ be two binary relations such that $\forall e_i, e_j \in E$, $e_i R_{\otimes} e_j \Leftrightarrow e_i$ and e_j are in the same part in $\mathcal{P}_1$ **or** in $\mathcal{P}_2$; $e_i R_{\oplus} e_j \Leftrightarrow e_i$ and e_j are in the same part in $\mathcal{P}_1$ **and** in $\mathcal{P}_2$. The transitive closure of $R_{\otimes}$ defines a partition which is $\mathcal{P}_1 \otimes_{\mathbb{P}} \mathcal{P}_2$ and $R_{\oplus}$ defines a partition which is $\mathcal{P}_1 \oplus_{\mathbb{P}} \mathcal{P}_2$.*

For example, $E = \{a, b, c, d, e, f, g\}$, $\mathcal{P}_1 = \{a, b\}, \{c\}, \{d, e\}, \{f, g\}$ and $\mathcal{P}_2 = \{a\}, \{b\}, \{c\}, \{d, \}\{e, f\}, \{g\}$. $\mathcal{P}_1 \otimes_{\mathbb{P}} \mathcal{P}_2 = \{a, b\}, \{c\}, \{d, e, f, g\}$.
Remind that a partition defines an equivalence relation. Thus $\otimes_{\mathbb{P}}$ can be viewed as the merging of the equivalence classes defined by $\mathcal{P}_1$ and $\mathcal{P}_2$.
Let's identify the generalization language $(\mathbb{L}, \leq_{\mathbb{L}}, \otimes_{\mathbb{L}})$ to describe the partitions of E, which is based on the generalization language $(L, \leq_L, \otimes_L)$ for E.
A partition of E is composed of subsets of E. Each of these subsets can be described with an element $l \in L$ with the description function d. Consequently, the description of a partition of E is described with a set $\mathcal{L}$ of elements of L. Thus, the description language for the partitions set is the powerset of L. Nevertheless, we only consider a subset of $\mathcal{P}(L)$. Indeed, we only need the set, noted $\mathcal{P}(L)_{\mathbb{P}(E)}$ of the subsets of L such that their instantiations are partitions of E. In the general case, the instantiation is only a coverset of E.

Definition 3 **($\leq_{\mathbb{L}}, \otimes_{\mathbb{L}}$).** *Let $(L, \leq_L, \otimes_L)$ be a generalization language. Let $\mathcal{L}_1$, $\mathcal{L}_2 \in \mathcal{P}(L)_{\mathbb{P}(E)}$. $\mathcal{L}_1 \leq_{\mathbb{L}} \mathcal{L}_2 \Leftrightarrow \forall l_1 \in \mathcal{L}_1$, $\exists l_2 \in \mathcal{L}_2$ such that $l_1 \leq l_2$.*
$\mathcal{L}_1 \otimes_{\mathbb{L}} \mathcal{L}_2$ is the set $\mathcal{L}_{\otimes}$ such that: $\mathcal{L}_{\otimes} = \big\{l \in L \ | l = d(P_1), P_1 \in merge(\{i(l_1) \cup i(l_2), l_1 \in \mathcal{L}_1, l_2 \in \mathcal{L}_2\})\big\}$

The definitions of $\leq_{\mathbb{L}}$ and $\otimes_{\mathbb{L}}$ are presented to keep mathematical formalism and are not used in practice. The description function $d_{\mathbb{P}}$ and the instantiation function $i_{\mathbb{P}}$ illustrate the method in a better way.

Definition 4 **(description of a partition $d_{\mathbb{P}}$).** *We define description function $d_{\mathbb{P}} : \mathbb{P}(E) \to \mathcal{P}(\mathbb{L})$ with: $\forall \mathcal{P} \in \mathbb{P}(E), d_{\mathbb{P}}(\mathcal{P}) = \big\{l \in L \mid l = d(P_1), P_1 \in merge(\{i \circ d(P), P \in \mathcal{P}\})\big\}$*

To find the description of a partition $\mathcal{P} \in \mathbb{P}(E)$, we first close the subsets P of E composing the partition with the closure $i \circ d$. This gives a coverset of E. Then, we merge transitively the subsets of this coverset which have at least one element e in common with the operator $\otimes_{\mathbb{P}}$. This gives a partition of E for which all subsets are closed elements of $i \circ d$, i.e. extents of the classical Galois lattice. Finally, we redescribe those elements with d.

Definition 5 **(instantiation of a subset of L $i_{\mathbb{P}}$).** *We define the instantiation function $i_{\mathbb{P}}(E)_L \to \mathbb{P}(E)$ with:$\forall \mathcal{P}_L \in \mathcal{P}(L)_{\mathbb{P}(E)}, i_{\mathbb{P}}(\mathcal{P}_L) = \big\{i(l), \ l \in \mathcal{P}_L\big\}$.*

The instantiation of a subset $\mathcal{P}_L \in \mathcal{P}(L)_{\mathbb{P}(E)}$, is simply the set composed with the instantiations of each element $l \in \mathcal{P}_L$. We can finally produce the partitions Galois lattice.

Theorem 1 (descriptions partitions Galois lattice).
$\mathcal{GL}_{\mathbb{P}}(\mathbb{P}(E),\subseteq_{\mathbb{P}},\otimes_{\mathbb{P}},\mathbb{P}(E)_L,\ \subseteq_{\mathbb{P}},\otimes_L,d_{\mathbb{P}},i_{\mathbb{P}})$ is a lattice called ***descriptions partitions Galois lattice****. Let $\mathbb{C}$ be the set of concepts of $\mathcal{GL}_{\mathbb{P}}$. Let $\mathcal{C} = (\boldsymbol{ext}, \boldsymbol{int}) \in \mathbb{C}$. The extension $\boldsymbol{ext}$ is a partition of E and the intention $\boldsymbol{int}$ is its description.*

With all the elements described above, the proof constits in showing that $(d_{\mathbb{P}} \circ i_{\mathbb{P}}, i_{\mathbb{P}} \circ d_{\mathbb{P}})$ defines a Galois connexion. Note that the same elements are needed to define the partitions Galois lattice compared to the classical Galois lattice: a set E and a generalization language $(L, \leq_L, \otimes_L)$. The classical properties of classical Galois lattices as a generalization space constrained with a language can be applied to the partitions Galois lattice: rules concerning regrouping objects, the possibility to deal with unknown objects, and a comparison between generalizations. The difference is that instead of considering the set of the regroupings of elements of E which can be expressed with the language L, we consider the set partitions of E which can be expressed with L.

Proposition 1 (partition agreeing with a language).
Let E be a set and $(L, \leq_L, \otimes_L)$ be a generalization language for E. A partition $\mathcal{P}(E)$ of E agreeing with the language L is a partition whose all the subsets can be expressed with an element $l \in L$. That is to say whose all the subsets are closed elements of $\mathcal{P}(E)$ with the closure $i \circ d$. The set of the extensions of the concepts of the description partitions Galois Lattice $\mathcal{GL}_{\mathbb{P}}$ is the set partitions of E agreeing with the language L.

Size of $\mathcal{GL}_{\mathbb{P}}$ The size of the $\mathcal{GL}_{\mathbb{P}}$ is equal to the number of partitions agreeing with L. Consequently, the size of the $\mathcal{GL}_{\mathbb{P}}$ is lower than the $|E|^{th}$ Bell number. The exact number of partitions agreeing with L is an open question. However, we can say that it's a function of the size of the classical $\mathcal{GL}$ size. In the worst case, the size of the $\mathcal{GL}$ is $2^{|E|}$. Then, the size of $\mathcal{GL}_{\mathbb{P}}$ is the $|E|^{th}$ Bell number, but in this case, L doesn't give any bias concerning the possible ways of regrouping the elements of E. This case is useless in our context. We can note that for each subset of E which can't be described with an element of L, the size of $\mathcal{GL}_{\mathbb{P}}$ is falling as a function of the number of elements of this subset.

4 Using the partitions Galois lattice for reinforcement learning

The set partition $\mathcal{P}^*$ of the environment's states induced by the quality function of optimal policies $Q^*(s,a)$ can be considered as observed section 2.2, as the goal of a reinforcement learning. We have just defined the descriptions partitions Galois lattice as space search for partitioning a set of objects, here S, the set of the environment's states. The space search being constrained with a language L_S, we have first to study the relationship between $\mathcal{P}^*$ and the possibility to express it with L_S. Then, we propose an algorithm to build $\mathcal{P}^*$.

4.1 Relationship between $\mathcal{P}^*$ and a generalization language L_S

Let an agent evolving in an environment formalized with a MDP $\mathcal{M}$ $\langle S,A,\psi,P,R\rangle$. The set S is described with the generalization language $(L_S, \leq_{L_S}, \otimes_{L_S})$.
Let's consider three different cases concerning the agreement of $\mathcal{P}^*$ with L_S. (1) *The partition $\mathcal{P}^*$ agrees with L_S*. In this case, each of the sets regrouping the states according to their optimal actions can be expressed with an element of the language $l \in L_S$. (2) *It exists at least one partition $\mathcal{P}$ agreeing with L_S* such that $\mathcal{P} \subseteq \mathcal{P}^*$. In this case, if we add the condition which is that each state $s \in S$ has a different description (the agent has a complete vision of its environment), it always occurs. Indeed, in the worst case, it's the partition of size $|S|$. Moreover, if the description language doesn't have any generalization properties that is to say the product of l_1, $l_2 \in L_S$ always gives the same element $l_\top$, then we have the classical case of reinforcement learning. Note that there can be more than one such partitions. (3) *There is no partition $\mathcal{P}$ agreeing with L_S such that $\mathcal{P} \subseteq \mathcal{P}^*$*. The agent is unable to distinguish states which should be considered different from the task point of view. We are in the case of Partially Observable MDP. We need for example, to use a memory based system [5] to separate the states.

4.2 A bottom-up Algorithm

The general framework we proposed: search for the partition $\mathcal{P}^*$, using the set partitions of S, can be implemented in several ways, depending on the chosen space exploration (bottom-up, top-down, breadth-first,...) and on the heuristics managing the imprecision of the current $Q(s, a)$ function.
We propose the bottom-up algorithm 1 which partitions the set of the states at a given step of the learning. The built partition is the most general, agreeing with L_S. The equivalence relation $\equiv^*$ doesn't always give a unique partition. Thus, the algorithm regroups in priority the states with the same value for their optimal actions.

Proof elements for the algorithm All the states are considered successively using a FIFO structure (variable `States` line 1). In the worst case, each state is added as a singleton to the resulting partition (line 7). This ensures that the final result is a cover of S. The lines 2 and 5 ensure that the elements are added to only one subset, ensuring that the result is a partition of S.
Line 3, the algorithm constructs a subset of S from a state and according to the equivalence relation $\equiv^*$, using te closure operator $i \circ d$. This ensures that the produced subset agrees with the language L. Then, we verify line 4 that the closure has only added equivalent elements, ensuring that the resulting partition is lower (using $\subseteq_{\mathbb{P}}$) than $\mathcal{P}^*$.

5 Experiment

We present an experiment of our algorithm on an academic problem to illustrate our method and to validate our theoretical approach with an implementation.

```
**Data**
• A generalization language $(L_S, \leq_{L_S}, \otimes_{L_S})$ for the states $S$ the current
estimation of $Q(s,a)$
**Result**
• A partition $\mathcal{P}$, based on $\equiv^*$, agreeing with the language $L_S$, such that $\mathcal{P}$ is
the most general possible, with grouping uppermost the states of $S$ with the
same optimal actions values.

  $Partition \leftarrow \emptyset$;
1 $States \leftarrow$ Sort $S$ according to the decreasing values of $Q(s, a^*)$;
  while $States \neq \emptyset$ do
    $s \leftarrow$ remove($States$);
    $Equivalents \leftarrow s_1 \in States \mid s_1 \equiv^* s$; $NewSubset \leftarrow \{s\}$;
    while $Equivalents \neq \emptyset$ do
2     $s_2 \leftarrow$ remove($Equivalents$);
3     $add \leftarrow i \circ d(NewSubset \cup \{s_2\}) - (NewSubset \cup \{s_2\})$;
4     if $\forall s_3 \in add$, $s_3 \equiv^* s$ then
        $NewSubset \leftarrow NewSubset \cup \{s_2\} \cup add$;
5       remove from $Equivalents(add \cup \{s_2\})$; remove from $States(add \cup \{s_2\})$;
7   $Partition \leftarrow Partition \cup \{NewSubset\}$;
  return $Partition$;
```

Algorithme 1 : Partitioning the set S according to $\equiv^*$ and agreeing with a generalization language $(L_S, \leq_{L_S}, \otimes_{L_S})$

The example consists in a grid-world of size (3×3) containing an *agent*, a *reward* and a *wall*. All the other tiles are *empty*. The walls can be moved, if there is no other behind, with the action *push*.
The task to be learned is to move as fast as possible to the reward tile. Each episode lasts until the agent reaches the reward tile or until it destroys it with a wall. The agent receives an reward of +1.0 if it reaches the reward tile. All other actions receive -0.1. There are 504 different states with 4 actions in each. *push* and *move* are different actions, $|\psi| = 2016$. The consequences of actions are deterministic and the environment is stationary.
The actions are described with: {*move west, move east, move north, move south, push west, push east, push north, push south*}. The language $(L_S, \leq_{L_S}, \otimes_{L_S})$ used to describe the states is an attribute-values language. The description of a state is composed with one attribute for each tile that the agent can see (see Figure 2). The partial order relation $\leq_{L_S}$ is defined by: $d(s_1) \leq_{L_S} d(s_2) \Leftrightarrow$ all the attributes of $d(s_1)$ are less than or equal to the corresponding one in $d(s_2)$ according to the join semi-lattice given Figure 2. The product $\otimes_{L_S}$ of two descriptions is the generalization of each attribute. The selection of actions and learning has been made with a classical *Q-Learning* algorithm.

Our algorithm allows the extraction of relevant description elements according to the task. Figure 3, shows an extract of the partition obtained after the convergence of the learning. Quantitatively, we need only 92 description elements

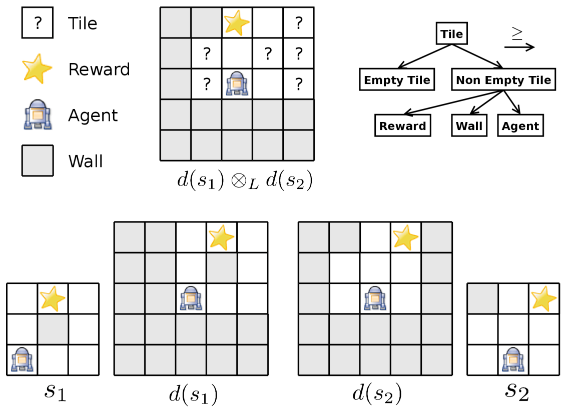


Fig. 2. A graphical example of two states, their descriptions and their product

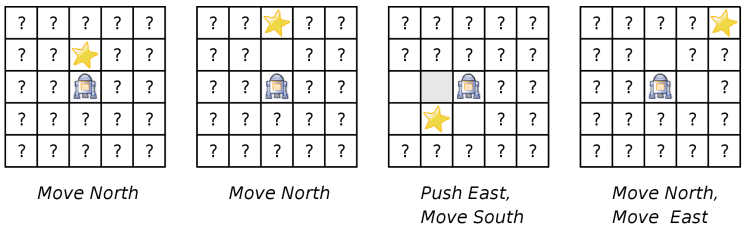


Fig. 3. Descriptions and optimal actions of 4 subsets among the 92 produced by partitionning the 504 environnement's states after convergence of the reinforcement learning

to describe the task instead of 504 initially. There are expected elements such as *being at one tile at the south of the reward* implies *move north* (example 1), the other tiles don't matter. We also find non intuitive rules (example 4).
Note that there are descriptions with the same optimal actions which are not grouped (example 1 and 2). This is because there are no language elements allowing the grouping of these descriptions without regrouping others, with different optimal actions. We could add, for example, the notion of relative position with the reward, either directly or using a knowledge data base.

6 Conclusion

First, we reminded the reader of the general principles in the use of a Galois lattice as a generalization space for a set of objects described with a generalization language. Then, we extended these results to define an original generalization space for the partitions instead of powerset, in particular by introducing the notion of partition agreeing with a language. We showed that, considering generalization as a partitioning, this structure can be used to generalize learning in reinforcement learning. Finally, we proposed a bottom-up algorithm which

through a reinforcement learning and a generalization language made it possible to produce general rules linking description elements of the environment and optimal actions.
This work considered principally the algebraical aspects. Consequently, we didn't treat two remaining important issues: "How to use the rules on unknown examples" and "How to deal with knowledge uncertainty".
Our future work will focus on three directions. First, a more formal link between our work and the relational reinforcement learning [9]. Secondly, an algorithmic improvement, studying the different ways of exploring the space search and the classical balancing between spatial and temporal complexity. Thirdly, we will consider the *exploration-exploitation* problem, i.e. using previously acquired knowledge or acquiring new knowledge and as a corollary, managing knowledge uncertainty. It remains a difficult problem in machine learning and in reinforcement learning in particular. In our case, it's transposed in a particular way as the concept reliability, already presented in [10]. We think that a relevant approach can consists in the explicit management of second order elements (states distribution, maximal reward,. . .) directly through the algorithms.

References

1. Watkins, C., Dayan, P.: Q-learning. Machine Learning **8** (1992) 279–292
2. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps : A framework for temporal abstraction in reinforcement learning. Artificial Intelligence **112**(1-2) (1999) 181–211
3. James, M.R., Singh, S.: Learning and discovery of predictive state representations in dynamical systems with reset. In: ICML 2004. (2004) 417–424
4. Munos, R.: Error bounds for approximate policy iteration. In: International Conference on Machine Learning ICML 2003. (2003) 560–567
5. McCallum, A.: Reinforcement Learning with Selective Perception and Hidden State. PhD thesis (1996)
6. Ravindran, B., Barto, A.G.: Smdp homomorphisms: An algebraic approach to abstraction in semi markov decision processes. In: IJCAI 2003. (2003) 1011–1016
7. Givan, R., Dean, T., Greig, M.: Equivalence notions and model minimization in markov decision processes. Artificial Intelligence **147**(1-2) (7 2003) 163–223
8. Dietterich, T.G.: State abstraction in maxq hierachical reinforcement learning. Artificial Intelligence Research (13) (2000) 227–303
9. Dzeroski, S., Raedt, L.D., Driessens, K.: Relational reinforcement learning. Machine Learning (43) (2001) 7–52
10. Ricordeau, M.: Q-concept-learning: Generalization with concept lattice representation in reinforcement learning. In Society, I.C., ed.: ICTAI 03. (2003) 316–323
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning : An Introduction. MIT press (1998)
12. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
13. Mitchell, T.M.: Machine Learning. McGraw-Hill (1997)
14. Liquière, M., Sallantin, J.: Structural machine learning with galois lattice and graphs. In Ed, M.K., ed.: ICML 1998. (1998) 305–313

Authors Index