Linköping Electronic Articles in
Computer and Information Science
Vol. 3(1998): nr 13

1st Order Logic Formal Concept Analysis: from logic programming to theory

Laurent Chaudron and Nicolas Maille

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1998/013/

Published on September 23, 1998 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. 3(1998): nr 13.
http://www.ep.liu.se/ea/cis/1998/013/. *September 23, 1998.*

This URL will also contain a link to the author's home page.

Abstract

In this paper, we analyze and define the introduction of 1st order logic in Formal Concept Analysis (FCA); the aims are both theoretical (as a complete model is needed) and applied (so as to improve expression power of FCA as a knowledge mining tool and the relevance of its results).

Our contribution consists in: i) the implementation of classical FCA in logic programming and the analysis of real cases, ii) the design of a complete 1st order FCA model, iii) the implementation of this 1st order FCA.

Keywords: Formal Concept Analysis, Knowledge Representation, Concepts Discovery, Rule induction, Logic Programming.

Authors' affiliation and address:

ONERA CERT
2 avenue Edouard Belin
BP 4025
F-31055 Toulouse Cedex 04 - FRANCE
Phone: +33 5 62 25 26 55 Fax: +33 5 62 25 25 93
{chaudron, maille}@cert.fr

Introduction

As far as knowledge information is concerned, the induction of concepts from data and information is a pivotal topic; generally, if numerical valuations (belief measures, preferences...) can be defined on the considered data, numerical or mixed methods can be directly used: rough sets [Paw91, SP98], Cartesian space model [IY98, IO98]... In some cases, requirements or accessibility constraints imply to rest only on symbolic attributes. Thus, more fundamental models and techniques are required; Formal Concept Analysis [GW96] is a good candidate for such a purpose. The present paper will contribute to the FCA approach following a bottom-up approach: from the classical FCA and its implementation in logic programming to the 1st order logic FCA theory.

This work represents the achievement of previous steps towards the introduction of first order logic in Formal Concept Analysis [Cha96]. In this paper, we present: the implementation of classical FCA in logic programming and the analysis of real cases in which knowledge discovery and exploration are needed, the design of a complete 1st order FCA model and the implementation of this 1st order FCA (the 1st order rule induction is not developed in the article).

Considering applied projects in which fundamental Formal Concept Analysis (say: **FCA**) was chosen so as to define and manage the knowledge involved in them, a concept lattice generator had to be implemented. For a sake of consistency with other requirements of the projects, this system, (called ***TC***, for *Treillis Conceptuel*) was written in a constraint logic programming language[1] (CLP). During the experiments, we discover that the 1st order capabilities of prolog in TC allowed contexts in which some "classical" propositional attributes were replaced by literals to be processed. Thanks to the prolog's capabilities, TC also offered some abductive features. But even a correct logic programming implementation does not constitute a correct proof neither a model of what could be a 1st order logic concept analysis. Hence, a closer investigation of the formal conditions for the introduction of first order logic into FCA had to be studied so as to ensure a relevant implementation and to offer new tools for knowledge management.

From a theoretical point of view, the relations between FCA and 1st order logic have been widely studied, especially in [Zic91]; our work is more focused on the formal prerequisites and the programming conditions of the definition of a consistent model of 1st order Logic FCA (say: **1LFCA**). This leads to search for 1st order corresponding operators to the fundamental ones in FCA: set union, set intersection, and the Galois connections (one must notice that nothing else than these three operators is needed to generate the FCA theory). This is the purpose of the Cube lattice model (section 3) thanks to which relevant definitions of 1LFCA can be formulated and implemented (section 4).

As it has been the case for FCA, the study of 1LFCA has created some results which may be of a theoretical interest for themselves, even if we also expect 1LFCA to offer more powerful exploration tools and in particular more suitable "numerical$\longrightarrow$symbolic" translations for the applications. Thus, the motivations of this work are both pragmatic and theoretical.

Note that the advantage of the "classical" FCA for knowledge processing, including concept emergence, data mining, rules discovery... is not the topic of the paper. Consequently we do not present any comparison

[1] more precisely, the program is coded in Prolog_3 ^{TM} *PrologIA*, a flexible constrained prolog

with other methods or models, neither discuss the interests and drawbacks of symbolic methods vs. numerical ones. FCA is considered here as an assumption in the context of the work.

1 FCA and Logic Programming

1.1 Formal Concept Analysis

Formal Concept Analysis (say: FCA) is a set-theoretical model for concepts that reflects the philosophical understanding of a concept as a unit of thought consisting in two parts: the *extend* which contains all the entities belonging to the concept, and the *intend* which is the collection of all the attributes shared by the entities. Based upon Galois connections FCA was designed by R. Wille in the 80ies, an introduction can be found in [DP90] and FCA theory and applications are now described in the reference book [GW96]. FCA is frequently used as a preprocessing tool for classification [CR96], but in our approach, we will stay closer to the original purpose of FCA. In FCA, the basic notion which models the knowledge about a specific domain is the formal *context* –described as binary relations between two finite sets– from which concepts and conceptual double hierarchies can be formally derived so as to form the mathematical structure of a lattice[2] with respect to a subconcept-superconcept relation. FCA is used for self-emergent classification of objects, detection of hidden implications between entities, construction of concept sequences, object recognition, aggregation of data and information, knowledge representation and analysis.

A context C is defined as a triple $(O, P, \triangleright)$ where O (objects or entities) and P (properties or attributes) are sets and $\triangleright$ is a binary relation on $O \times P$ which indicates whether an entity has an attribute or not.

The formal *concepts* set F can be derived from this context as follows: For $A \subset O$ and $B \subset P$, let us define the Galois connections: $A' = \{p \in P | (\forall o \in A)\ o \triangleright p\}$; $B' = \{o \in O | (\forall p \in B)\ o \triangleright p\}$. Thus, A' is the set of the attributes that are common to all the entities in A and B' is the set of the entities possessing their attributes in B. These Galois connections verify all the usual properties of the duality; in particular the following relation $A \subset A''$ stands for any subset of the context (consequently $A' = A'''$). A concept is then a pair (A, B) such that $A \subset O,\ B \subset P,\ A' = B$ and $B' = A$. A is the extend of the concept and B the intend.

For concepts (A_1, B_1) and (A_2, B_2) the hierarchical *subconcept - superconcept* relation is formalized by : $(A_1, B_1) \leq (A_2, B_2) \iff A_1 \subseteq A_2 (\iff B_2 \subset B_1)$.

The set of all the concepts of the context $(O, P, \triangleright)$ together with this order relation is a complete lattice which is called the *concept lattice*. Therefore for every set of concepts there exists a unique largest subconcept (the *infimum*) and a unique smallest superconcept (the *supremum*). Supremum and infimum are respectively given by :

$$\bigvee_{j \in J} (A_j, B_j) = ((\bigcup_{j \in J} A_j)'', \bigcap_{j \in J} B_j),$$

[2] given two internal operators $\sqcap$ (infimum) and $\sqcup$ (supremum) on a set E, $(E, \sqcap, \sqcup)$ is a lattice, iff$_{def}$: $\sqcap$ and $\sqcup$ are idempotent, associative, commutative and they verify the absorption law $x \sqcap (x \sqcup y) = x$ and $x \sqcup (x \sqcap y) = x$. A lattice is an ordered set: the relation $\leq$ defined on E as: $(x \leq y) \leftrightarrow_{def} (x \sqcap y) = x$ is an order relation for which $\sqcap$ and $\sqcup$ represent the greatest lower bound and the least upper bound [Bir40].

$$\bigwedge_{j\in J}(A_j, B_j) = (\bigcap_{j\in J} A_j, (\bigcup_{j\in J} B_j)'').$$

Let us consider a simple example given by the following context C_0:
$8 : (\{\}, \{Pro1, Pro2, Pro5, Pro6, Pro3, Pro4\})$
$7 : (\{Obj3\}, \{Pro1, Pro3, Pro5\}))$
$6 : (\{Obj2\}, \{Pro2, Pro3, Pro4, Pro5\})$
$5 : (\{Obj1\}, \{Pro1, Pro2, Pro5, Pro6\})$
$4 : (\{Obj3, Obj2\}, \{Pro5, Pro3\})$
$3 : (\{Obj2, Obj1\}, \{Pro5, Pro2\})$
$2 : (\{Obj3, Obj1\}, \{Pro5, Pro1\})$
$1 : (\{Obj1, Obj2, Obj3\}, \{Pro5\})$

This concept lattice can be represented by the Hasse-diagram of Fig.1. As only 3 entities are concerned, this example is very simple. Let us see how to exploit the knowledge captured by F_0 and first how to read the diagram.

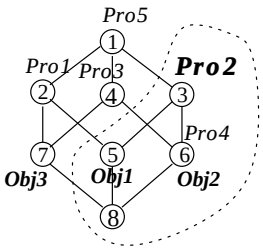

Figure 1: The Concept Lattice F_0

On such a diagram, an element labeled by an entity Obj_i represents the concept with the smallest set containing Obj_i, and an element labeled by a property Pro_j represents the concept with the smallest set containing it. Hence the interpretation of the concept lattice is easy: a given concept ⓝ inherits all the properties which are linked above it in the diagram and ⓝ is constituted of all the entities which are linked below it. For example, it is clear to find again in the diagram that property Pro_5 is verified by all the entities as it is the label of the top element ① in the lattice. If we focus on Pro_2, the dotted line describes all the concepts verifying property Pro_2. It is easy to notice that concept ③ labeled by Pro_2 is below concept ① which is labeled by Pro_5. This captures a contextual implication: $Pro_2 \rightarrow Pro_5$; many contextual implications are deducible from a concept lattice [GD86]. Our logic programming approach allows any deductive or inferential extrapolation from the context to be computed.

Moreover, in any concept lattice F_n, if $\bot$ and $\top$ denote the universal elements (in our example F_0, $\bot$ corresponds to ⑧ and $\top$ corresponds to ①), an interval such as $[Obj_i, \top]$ represents all the concepts containing Obj_i; and an interval $[\bot, Pro_j]$ represents all the concepts verifying Pro_j. Given an entity Obj_i and a property Pro_j the set of all the entities related with Obj_i about Pro_j is interval $[Obj_i, Pro_j]$. Converse properties and laws can also be derived: for the sake of protection of information one may search for concepts not containing a given entity or a precise property: the converse intervals can be considered as solutions $[\bot, Obj_i[$ or $]Pro_j, \top]$.

All these considerations are classical in domains such as social sciences or data analysis [Stu95]. Their application to data mining or knowledge discovery in large databases is amplified by the logic programming capabilities of our system.

1.2 FCA in prolog

A Prolog program, called ***TC*** was designed so as to implement the Galois connections and the function which associates to each context C its Concept Lattice L: $C \longrightarrow L$. The context and the concept lattice generated by TC can be considered as a global knowledge base $C \cup L$ on which knowledge exploration experiments are performed.

Remark: It must be noticed that the computing time is **not** a critical problem in our projects. Moreover, the lattice must be completely developed and is completely developed. This must not be considered as curious: indeed, FCA is generally known through its use in the frame of data analysis as a prefilter. In these approaches the lattice is partially computed for a sake of time computing. Here, our interest of FCA lays in its origin: how to discover the structure and all the links within an apparently amorphous set of knowledge. Such an approach is required when a very new domain is explored, and this is the case in two of our applications: interoperability and space data analysis.

The choice of CLP aims at allowing both all classical databases functionalities and reasoning capabilities to be available. Of course, the complexity of the algorithm is not good (non-polynomial) but this has no incidence as the cardinal of each set of considered entities is small (< 20). The algorithm is based on a simple saturation method (the atoms are first computed and the lattice is generated by applications of the supremum operator until the top element appears).

Let us have a look at the example [3] of a fictitious context C_1 in which the properties are listed in raws :

	Claudia	Naomi	Cindy
movie-star	×		×
netizen			×
GVersace		×	
blue_eyes	×		
brown_eyes		×	×
P_1	×		
P_2		×	
P_3			×

In TC, the context is coded with a 2-argument predicate "Context":
$Context(name\_of\_the\_entity, < set\_of\_its\_attributes >)$
(the brackets are used as set marks). Let us suppose that the three last properties of this context, which are denoted with non-informative identifiers, can be defined as triples of values: $P_1 =< 95, 62, 92 >$, $P_2 =< 86, 56, 82 >$, and $P_3 =<??, 56, ?? >$, where "??" represents an unknown value. C_1 becomes:
$Context(Claudia, < movie\_star, < 95, 62, 92 >, blue\_eyes >)$
$Context(Naomi, < GVersace, < unknown1, 56, unknown2 >, brown\_eyes >)$
$Context(Cindy, < netizen, movie\_star, < 86, 56, 82 >, brown\_eyes >)$

One may notice that the unknown parameters $<??, 56, ?? >$ (last column of the table) of $Naomi$ are coded as constants: $< unknown1, 56, unknown2 >$. In TC, a saturation algorithm generates the concept lattice which is coded with a 2-argument predicate "Concept":

[3] the authors precise that this humourless simplified example - induced during a workshop - does intend to be respectful.

$Concept(number, << set\_of\_entities >, < set\_of\_attributes >>)$
thus the concept lattice L_1 of C_1 is:
$Concept(7, <<>, < movie\_star, < 95, 62, 92 >, blue\_eyes, GVersace,$
$< unknown1, 56, unknown2 >, brown\_eyes, netizen, < 86, 56, 82 >>>)$
$Concept(6, << Naomi >, < GVersace, < unknown1, 56, unknown2 >,$
$brown\_eyes >>)$
$Concept(5, << Cindy >, < netizen, movie\_star, < 86, 56, 82 >, brown\_eyes >>)$
$Concept(4, << Claudia >, < movie\_star, < 95, 62, 92 >, blue\_eyes >>)$
$Concept(3, << Naomi, Cindy >, < brown\_eyes >>)$
$Concept(2, << Claudia, Cindy >, < movie\_star >>)$
$Concept(1, << Claudia, Naomi, Cindy >, <>>)$

Let us look at the Hasse's diagram of L_1, labeled with the classical manner but on which only the shared attributes (i.e. at least by two entities) are represented:

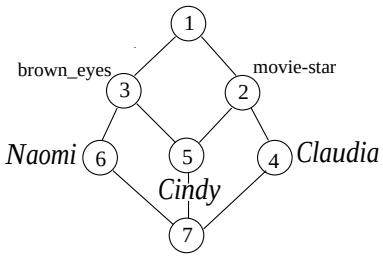

Figure 2: The FCA lattice L_1

Of course the unknown feature $<??, 56, ?? >$ of *Naomi* could be coded in a more relevant way regarding the first order capabilities of Prolog: $< V1, 56, V2 >$ – where each V_i is a variable, with respect to Prolog_3^{TM} syntax – could be a good means to represent the partial ignorance on the two parameters. Moreover, the color of the eyes could be capture in a quasi-predicative way such as: $eye(blue)$ and so on. Of course, the context is lightly modified as everything happens as if the attribute *eye_brown* was split into two attributes $eye(brown(dark))$ and $eye(brown(light))$. This detail will be studied in the sequel. Thus, it is possible to refine the context as follows, C_2:

$Context(Claudia, < movie\_star, < 95, 62, 92 >, eye(blue) >)$
$Context(Naomi, < GVersace, < V1, 56, V2 >, eye(brown(dark)) >)$
$Context(Cindy, < netizen, movie\_star, < 86, 56, 82 >, eye(brown(light)) >)$

From this new context C_2, TC generated what we may call again a "concept lattice" L_2 but the characteristics of which are program-dependent (the variable identifiers X_{62}... are automatically given by Prolog_3^{TM}):

$Concept(7, <<>, < movie\_star, < 95, 62, 92 >, eye(blue), GVersace,$
$< X_{62}, 56, Y_{62} >, eye(brown(dark)), netizen, eye(brown(light)) >>)$
$Concept(6, << Naomi >, < GVersace, < X_{68}, 56, Y_{68} >, eye(brown(dark)) >>$
)
$Concept(5, << Cindy >, < netizen, movie\_star, < 86, 56, 82 >,$
$eye(brown(light)) >>)$
$Concept(4, << Claudia >, < movie\_star, < 95, 62, 92 >, eye(blue) >>)$
$Concept(3, << Naomi, Cindy >, << 86, 56, 82 >>>)$
$Concept(2, << Claudia, Cindy >, < movie\_star >>)$

$Concept(1, << Claudia, Naomi, Cindy >, <>>)$

From a structural point of view, the diagram is labeled as follows:

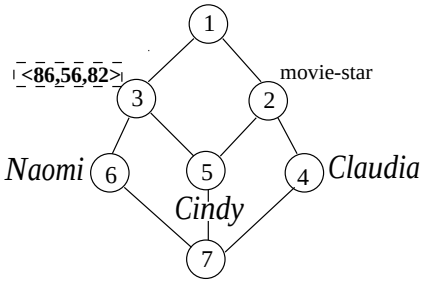

Figure 3: The "concept lattice" L_2

In the concept lattice L_2, one can notice that the natural unification mechanism of Prolog, combined with the FCA generation algorithm allowed TC to make an abduction on concept ③: *Naomi* is supposed to inherit the properties of the triple $< 86, 56, 82 >$ of *Cindy* as the two terms are syntactically unifiable. Such an abduction tool is fruitful but it must be controlled by a correctly defined model. Indeed, the unification mechanism is algorithm dependent in TC and no global definition can justify if $< 86, 56, 82 >$ must belong to the top concept ① or not. Moreover, looking closely at *Naomi* and *Cindy*, one can imagine that, as far as knowledge representation is concerned, the features $eye(brown(dark))$ and $eye(brown(light))$ would have to be considered as having something in common, such as: $eye(brown(??))$. This will be defined in section 3 after a survey of the application of TC on a real case.

1.3 Application: Interoperability Analysis

This application is related to the design of distributed information systems: given a set of information systems, they are supposed to operate together in order to achieve peace maintenance or rescue missions, the problem is how to define and determine all the subgroups of systems, sharing the same capabilities so as to understand and to better manage the whole set of systems? Such systems are supposed to co-operate, hence they are supposed to understand each other: this is the informal definition of interoperability. A subset of the systems is supposed to operate correctly (i.e. to interoperate) if the systems share the required capabilities. Thus, FCA appears to be a good tool for such an interoperability analysis.

The knowledge required for a mission is supposed to be simplified so as to be captured by a set of attributes. Such an attribute may represent a specialty, a feature or a capability of a system (e.g. medical material database access, languages translations, transmission functionality). Thus, from a strict semantic point of view, the systems and the attributes define the context of interoperability [CB97].

For example, the following context describes the features of five systems categories. For the sake of confidentiality, the real names and attributes have been replaced by fictitious ones: VEH is an on-board information system of a vehicle, TRANS is a transmission system, XFLR6 is a mobile surgery cell, SYS33 is a local information system and SYSF99 is a global rescue management system. The formal interoperability analysis leads to a

better understanding of the inner semantic relations between the systems for their missions; the common semantic zones are thus expressed in terms of concepts. Thanks to equivalence relations between the features, the context C_3 summarizes a four-page description file:

$Context(VEH, \{omouv(meca)\})$
$Context(TRANS, \{opo(ue), opo(odb), omouv(ue), cr(sit2), cr(tir), dem(tir)\})$
$Context(XFLR6, \{opo(gu), cr(sit), dem(tir)\})$
$Context(SYSF99, \{opo(gu), opo(SYS33), opo(rens), opo(odb), cr(sit)\})$
$Context(SYS33, \{opo(gu), opo(SYS33), opo(ue), opo(rens), opo(odb),$
$omouv(ue), cr(sit), cr(sit2), prev(tirnucami), dem(tir), omouv(meca)\})$

It can be noticed again that the attributes of the systems are expressed through first order logic properties. The Concept Lattice L_3 derived from C_3 by TC is constituted of 11 concepts defining totally interoperable groups (see fig. 3).

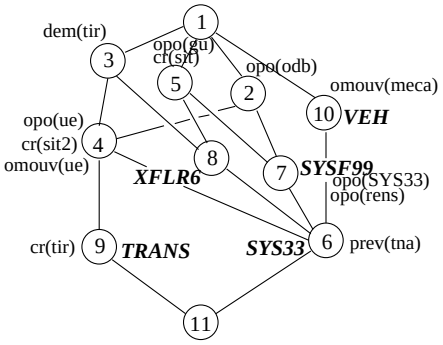

Figure 4: The Interoperability Concept Lattice L_3

The knowledge base $C_3 \cup L_3$ is used so as to look for contextual dependencies between either systems or attributes. The induction of the context-based rules is given by the generic frame of [GD86] widely developed in *ConImp* by Burmeister since [Bur87]. The axiom's frame is based on the classical law: for every subset A of properties of the context, $\models (A \rightarrow (A'' - A))$. For instance, a question may be: "from the present context, what are the properties that are necessarily deduced when systems have to operate by sharing capabilities on rescue operations planning ($opo(x)$) and medical staff movements ($omouv(y)$)?". The transaction consists in searching the Galois's closure of $\{opo(x), omouv(y)\}$. In context C_3, the following interdependency is proved:

$\{opo(odb), omouv(ue)\} \rightarrow \{dem(tir), cr(sit2), opo(ue)\}$

This represents a first step to the exploration logical inductive capabilities of TC.

Preliminary results: For each set of interoperating systems concerned, FCA –and more precisely TC– has allowed the totally interoperable groups to be defined and managed. The most important work which is currently under study is the development of the CLP induction capabilities of TC so as to i) determine cross requirements, ii) prevent dead-locks design and iii) build methodologies. Of course, the extension of TC to $TC1$ will be integrated in these studies (these results will not be developed in the sequel of the paper).

Back to the technical questions in FCA, it is clear again that a CLP approach in FCA offers quasi-1st order characteristics considering that the attributes are defined as: $pred(arguments)$ in which each argument is a

constant. But as $opo(odb)$ and $opo(ue)$, are considered as totally different attributes by TC, it does not constitute an actual 1st order FCA model. A desired result would be the "natural" induction of a $\{opo(\mathbf{x})\}$ literal. As in the simple example of the context C_2, a anti-unification mechanism is here needed. In next section, the Cube Lattice model gives the theoretical prerequisites for such requirements.

2 The cube model

The purpose of the cube model is to build, in a 1st order logic frame, the same kind of structure than the implicit logical propositional lattice $(L_0, \cup, \cap)$ used by FCA[4], and in which the partial order $\subset$ is consistent with the logical implication $\rightarrow$. The cube model is based on a classical first order language (*Const, Var, Funct, Pred*) whose set of terms *Term* is the functional closure of *Const* $\cup$ *Var* by *Funct*. This model has already been used for knowledge representation in intelligence systems and cooperative systems [CCMT97]. The elementary properties are represented by literals and the elements of their power set $\boxed{\mathcal{C}}$ are called $\boxed{\text{logical cubes}}$[5]; they are interpreted as the conjunction of the literals. Cubes play a dual role besides the classical clauses, and by default their variables are existentially quantified.

As far as knowledge representation is concerned, we would like the order relation induced on $\mathcal{C}$ to capture the intuitive notion of "information enrichment". But such an "enrichment" can be obtained via different means: quantity of information, precision of terms, logical dependency. Example: $\{cr(sit), opo(gu)\}$ is more informative than $\{opo(gu)\}$ for the number of literals is higher; and $\{cr(sit)\}$ is more informative than $\{cr(x)\}$ for a sake of precision. Unfortunately, the combination of both intuitive criteria may lead to a contradiction: $\{cr(x), opo(gu)\}$ cannot be consistently compared to $\{cr(sit)\}$.

The previous cases highlight the need for sound definitions to the intuitive concepts of union and intersection of two finite information sets in accordance to the following requirements: the infimum has to capture the common features (while giving more information than the empty set frequently generated by the unification rule); the supremum has to cope with the contradictory criteria: quantity/precision of the information (while giving a more synthetic result than the set union). Such purposes are achieved thanks to an algebraic approach whose results are also examined from a logical point of view. If the definition of the supremum and the infimum operators can be supported by the set union and intersection in the propositional calculus frame, first order logic needs more sophisticated tools. We use the subsumption and reduction operators defined by Plotkin [Plo70], but also adopt the approach of [Hue76] and [LMM87] which allows a lattice on the terms algebra to be defined properly thanks to the *anti-unification* operator.

Example: $p(x, g(y, b))$ is the anti-unified literal of $p(a, g(a, b))$ and $p(1, g(b, b))$.

In fact, anti-unification allows the infimum to generalize the terms so as

[4] contrary to what some authors seem to think the set operators are the only basic properties required so as to design FCA theory, or, in other words, Galois connections, cf. "polarities" [Bir40] pp 122–126.

[5] "cube" is the name which was used for the first time by A. Thayse [Tc89]. The denotator "product"was previously used –but not defined– in 1975 by Vere [Ver75]

to properly enrich the set intersection on the cubes. The result of the anti-unification of two cubes c_1 and c_2 is defined as the union of the anti-unification of every couple (l_1, l_2) based on the same predicate name and such that l_1 belong to c_1 and l_2 to c_2.

Conversely, it is essential to reduce the mere set union of cubes in order to define the supremum so as to discard redundancies and to guarantee the verification of the lattice's absorption law. Based upon the same approach than for the anti-unification, the definition of such a reduction relies on the class of the substitutions on terms.

Definition: A cube C is reducible if there exists a substitution θ such that $C\theta \subsetneq C$. It is easy to see that:

Proposition: An irreducible reduction of C always exists and is unique up to variable renaming. It will be noted *reduc(C)*.

Examples: It is clear that *reduc({a(x),a(1)}) = {a(1)}*, but *reduc({a(1,x),a(y,2)}) = {a(1,x),a(y,2)}*. If $\mathcal{C}^r$ denotes the subset of all the irreducible cubes of C, it is possible to define constructive operators on C^r:
Definition: Let c_1 and c_2 belong to $\mathcal{C}^r$. The infimum and supremum operators $\cup_c$ and $\cap_c$ are defined as:
$c_1 \cup_c c_2 =_{def} reduc(c_1 \cup c_2)$;
$c_1 \cap_c c_2 =_{def} reduc[anti\text{-}unif(c_1, c_2)]$.

Theorem $(\mathcal{C}^r, \cup_c, \cap_c)$ is a lattice.
Indeed, this lattice on cubes corresponds to the lattice on clauses defined in [Plo70].

Definition of the infimum (resp. supremum) of a set: $\cap_c$ (resp. $\cup_c$) being an associative operator, the infimum (resp. supremum) of a set S of cubes will be denoted by $\cap_{c\ c_i \in S}\ c_i$ (resp. $\cup_{c\ c_i \in S}\ c_i$).

Remark: The extension of Huet's recursive anti-unification algorithm to the cubes lends itself well to a CLP implementation. Thus our approach proposes a more easily computable definition of the lattice than the initial definition of Plotkin; it has also been implemented in Prolog_3™.

If we consider again the common features of *Naomi* and *Cindy* we are now ready to determine the supremum and infimum of the two cubes:
$<< Var1, 56, Var2 >, eye(brown(dark)) >$ for *Naomi*,
and $<< 86, 56, 82 >, eye(brown(light)) >$ for *Cindy*[6].

In the following prolog trace, SUP and INF are the ternary prolog predicates which code the supremum and infimum (first line: user's query, second line: prolog's answer) :
$SUP(<< Var1, 56, Var2 >, eye(brown(dark)) >, << 86, 56, 82 >, eye(brown(light)) >, S)$;
$\{\boldsymbol{S} =< eye(brown(dark)), < 86, 56, 82 >, eye(brown(light)) >\}$

$INF(<< Var1, 56, Var2 >, eye(brown(dark)) >, << 86, 56, 82 >, eye(brown(light)) >, I)$;
$\{\boldsymbol{I} =< eye(brown(Var1)), < Var2, 56, Var3 >>\}$

Thus, the requirements of unification (resp. anti-unification) for the supremum (resp. the infimum) are verified. They will represent the core

[6]the reader must notice that a particular syntax must be adopted so as to capture the variables in the cube lattice sense and to distinguish them from the Prolog_3™ ones. A cube lattice variable is a identifier begining with $Va...$

process of the 1st order Galois connections.

Remark: the program respects the usual logic programming requirements: the range of a variable is bounded in the cube in which it appears. Thus, in the previous example, variable $Var1$ inside the cube I is not linked to $Var1$ which appeared in the query. This requirement will also be verified within the complete 1LFCA model (section 4).

Propositions:
• If we denote $\boxed{\leq_c}$ the order relation induced by the lattice structure on $\mathcal{C}^r$, thus $\boxed{\leq_c}$ is identical to the subsumption: $c_1 \leq_c c_2$ iff there exists a substitution θ such that $c_1\theta \subseteq c_2$.
• $(\forall c_1, c_2 \in \mathcal{C}^r)(c_1 \leq_c c_2)$ iff $\models (c_2 \rightarrow c_1)$. It is worth noticing that the design of the lattice structure on the set of logical cubes does not require the *a priori* definition of a partial order on $\mathcal{C}^r$. Moreover, $\mathcal{C}^r$ is identically ordered by $\leq_c$ and $\leftarrow$; this guarantees the completeness of the approach.

Note: The Cube model is widely used in a complete knowledge discovery multi-agent methodology based on the philosophical work of Lakatos [TC96, CFMT97][7]; this approach is close to [MA95].

3 1st order FCA

We are now ready to define a new formal concept analysis model in which the attributes of the entities are captured by a set of literals from a first order logical language instead of literals from a propositional language. Thus an entity of the context is characterized by a logical cube.

Let a first order context be given by a set O of entities, the set P of all the reduced cubes generated with the first order logical language and a map $\square$ from O to P. Each entity o in O has one and only one image by $\square$ in P. This image $p = \square(o)$, represents the set of properties of o.

Let A be a finite set of entities and B a literal: $A \subset O$ and $B \in P$. The Galois connections are defined as follows (one can easily come back to the definitions of FCA by replacing the usual set union and intersection):

$$A' =_{def} \cap_{\mathbf{c}\ o_i \in A}\ \square(o_i)$$
$$B^\circ =_{def} \{o_i \in O / B \leq_c \square(o_i)\}.$$

A $\boxed{\text{first order concept}}$ is a pair (A, B) such that:
$A \subset O,\ B \in P,\ A' = B$ (up to variable renaming) and $B^\circ = A$.

For concepts (A_1, B_1) and (A_2, B_2) the hierarchical *subconcept - superconcept* relation is formalized by:
$(A_1, B_1) \leq (A_2, B_2) \Longleftrightarrow A_1 \subseteq A_2 (\Longleftrightarrow B_2 \leq_c B_1)$.

The set $\boxed{\mathcal{L}^1}$ of all the 1st order concepts of the context $(O, P, \square)$, together with this order relation, is a complete lattice which is called the $\boxed{\text{first order concept lattice}}$. Therefore for each set of concepts there exist a unique largest subconcept (the *infimum*) and a unique smallest superconcept (the *supremum*). The supremum and infimum are respectively given by:

$$\sqcup_{j\in J}(A_j, B_j) = ((\bigcup\nolimits_{j\in J} A_j)'^\circ, \cap_{\mathbf{c}\ j\in J}\ B_j),$$

$$\sqcap_{j\in J}(A_j, B_j) = (\bigcap\nolimits_{j\in J} A_j, (\cup_{\mathbf{c}\ j\in J}\ B_j)^{\circ\prime})$$

[7] see also ECAI'98 Workshop on Conflicts:
`http://www..cogs.susx.ac.uk/ecai98/tw/W16.html`

Theorem $(\mathcal{L}^1, \sqcup, \sqcap)$ is a lattice.

Proof: the native formal concept analysis constructions in which attributes map with literals of a propositional language can be followed substituting $\cup_c, \cap_c$ *and* $\leq_c$ *by* $\cup, \cap$ *and* $\subseteq$. *The proof of the lattice structure relies only on the classical properties of infimum, supremum and order relation operators of the frame lattice, it is sufficient to adapt the classical ones (e.g. [DP90] p.224).*

A new program: $TC1$:
The TC algorithms were adapted to the 1LFCA definitions with respect to the Cube lattice program, this gave the first 1st order logic concept lattice generator called $TC1$. It has allowed to consider C_2 as a full first order context[8]:

$Context(Claudia, < movie\_star, < 95, 62, 92 >, eye(blue) >)$
$Context(Naomi, < GVersace, < Var1, 56, Var2 >, eye(brown(dark)) >)$
$Context(Cindy, < netizen, movie\_star, < 86, 56, 82 >, eye(brown(light)) >)$

The first 1st order concept lattice is:

$Concept1(7, <<>, < netizen, eye(brown(light)), GVersace, < 86, 56, 82 >, eye(brown(dark)),$
$movie\_star, < 95, 62, 92 >, eye(blue) >>)$
$Concept1(6, << Naomi >, < GVersace, < Var1, 56, Var2 >,$
$eye(brown(dark)) >>)$
$Concept1(5, << Cindy >, < netizen, movie\_star, < 86, 56, 82 >,$
$eye(brown(light)) >>)$
$Concept1(4, << Claudia >, < movie\_star, < 95, 62, 92 >, eye(blue) >>)$
$Concept1(3, << Cindy, Naomi >, < eye(brown(Var1)), < Var2, 56, Var3 >>>)$
$Concept1(2, << Cindy, Claudia >, < eye(Var1), < Var2, Var3, Var4 >,$
$movie\_star >>)$
$Concept1(1, << Cindy, Naomi, Claudia >, < eye(Var1),$
$< Var2, Var3, Var4 >>>)$

One must recall that the variables in a cube are existentially quantified, the range of a variable is bounded inside each cube. This explains why different cubes may contain the same variable identifier (which is given by the TC_1) without confusion.

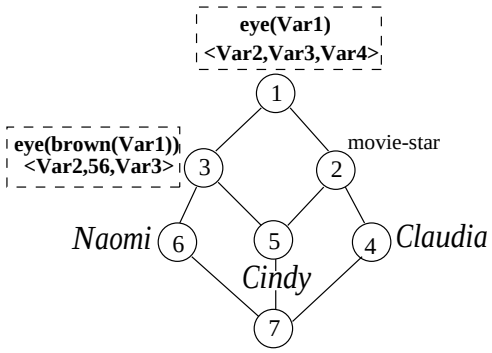

Figure 5: A First Order Concept Lattice L_4

This time, and contrary to the uncertain situation of concept lattice

[8]recall: each Var_i is a $TC1$ variable

L_2 (fig. 2), the common properties of Naomi and Cyndi is sumerized in the cube: $< eye(brown(Var1)), < Var2, 56, Var3 >>$, which is more relevant than the direct induction performed in L_2 as the concept shows the more cautious and sensible set of shared properties. Naomi may, but she is not obliged to, get exactly the same values properties $< 86, 56, 82 >$ than Cindy.

Back to the formal interoperability analysis, with the same context C_3, $TC1$ determines two new concepts ② ' and ⑤ ' in figure 5. They represent the generalized attributes $opo(Var1)$, $cr(Var2)$ and $omouv(Var3)$ which were searched for in the first implementation in TC.

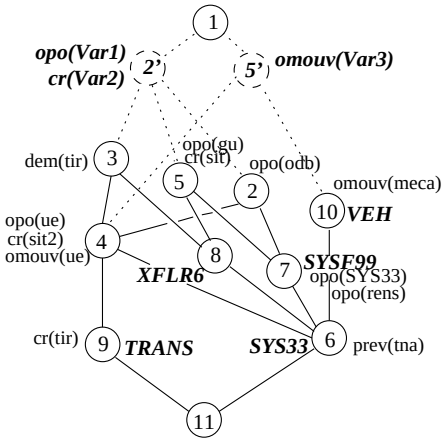

Figure 6: The First Interoperability Lattice L_5

The extension of TC to $TC1$ is a recent result and $TC1$ has to be more intensively experimented. In particular, the powerful cubes tool will allow inner logical constraints thanks to variables to be taken into account: in the cube $< foo(Var1, Var2), bar(Var2, trick) > foo$ and bar are linked through variable $Var2$. Furthermore, a comparison between the 1LFCA capabilities and those of proximate models such as logical scaling [Pre97] has been launched.

From an applicative point of view, another specific project related to space data analysis has been studied; its empirical results are currently extended thanks to 1LFCA and the developments are under study. Indeed, the favorite domain of FCA and 1LFCA is the symbolic knowledge, but thanks to CLP and the experiments with $TC1$, many improvements are expected for the analysis of large numerical databases. Finally, many theoretical conjectures (e.g. "a FCA concept lattice is always a sub-lattice of its 1LFCA one"?) are still to be determined and proved.

4 Conclusion

The first step towards 1st order Formal Concept Analysis was the implementation of classical FCA algorithms in CLP (Constraint Logic Programming). With this system, knowledge exploration experiments and knowledge discovery methodologies have been studied in various projects (interoperability, data analysis).

From a theoretical point of view, the integration of the *Cube* model – a lattice structure on conjunctions of 1st order literals – in FCA constitutes the second step towards a complete 1st order FCA.

The next step will be the extension of the Cube model to a Constraint-Cube model – a lattice of constrained cubes – so as to offer a powerful numerical and symbolic data analysis and knowledge discovery. This should also help to reduce the complexity of the determination of the concept lattice thanks to the clustering of similar attributes.

The current studies focus on the development of the 1st order rule generation module of $TC1$ related to the classical approaches of the ILP[9] community and its application to a new domain: knowledge discovery in human/machine interaction through the analysis of pilot/plane human factors knowledge bases.

Abbreviations
FCA: Formal Concept Analysis
1LFCA: First Order Formal Concept Analysis
CLP: Constraint Logic Programming
TC: (Treillis Conceptuel) the prolog system for FCA
TC1: the prolog system for 1LFCA

References

[Bir40] G. Birkhoff. *Lattice Theory.* ACM, 1940.

[Bur87] Peter Burmeister. *Programm zur Formalen Begriffsanalyse einwertiger Kontexte.* TH Darmstadt, 1987. English Version.

[CB97] L. Chaudron and M. Barès. Interoperability of systems: from distributed information to cooperation. In *15th IJCAI'97 Workshop: "IA in Distributed Information Networking"*, Nagoya, Japan, August 22nd 1997.

[CCMT97] L. Chaudron, C. Cossart, N. Maille, and C Tessier. A purely symbolic model for dynamic scene interpretation. *International Journal on Artificial Intelligence Tools*, 6(4):635–664, 1997.

[CFMT97] L. Chaudron, H. Fiorino, N. Maille, and C. Tessier. From Lakatos to lattices: a semantic control of cooperative dialogues. In *IJCAI'97, Intl. Join Conf. On Artificial Intelligence, Workshop "Collaboration, Cooperation and Conflicts in dialogue systems"*, Nagoya, Japan, August 23nd 1997. (extended version to appear in: "Computational Conflicts", J.Mueller and R.Dieng eds.).

[Cha96] L. Chaudron. First steps towards first order logic in formal concept analysis. In *Intl. Conf. on Conceptual Knowledge Processing*, Darmstadt, Germany, 28Feb.-1Mar. 1996.

[CR96] C. Carpineto and G. Romano. A lattice conceptual clustering system and its application to browsing retrievial. *Machine Learning*, 24:95–122, 1996.

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

[GD86] J-L Guigues and V. Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Math. Sci. Hum.*, 95:5–18, 1986. In French.

[9] Inductive Logic Programming

[GW96] Bernhard Ganter and Rudolf Wille. *Formale Begriffsanalyse: Mathematische Grundlagen.* Springer–Verlag, Berlin–Heidelberg, 1996. In German.

[Hue76] G.P. Huet. *Résolution d'équations dans des langages d'ordre 1,2,...,ω.* PhD thesis, Université de Paris VII, 1976. Thèse d'État (in French).

[IO98] M. Ichino and Y. Ono. A new feature selction method to extract functional structures from multidimensional symbolic data. *IEICE transactions on information and systems*, E81-D(6):556–564, June 1998.

[IY98] M. Ichino and H. Yaguchi. *Data Science, Classification and related Methods*, chapter Symbolic Pattern Classifiers Based on the Catesian System Model, pages 358–369. Springer-Verlag, 1998.

[LMM87] J-L. Lassez, M.J. Maher, and K. Marriott. *Foundations of Deductive Databases and Logic Programming*, chapter Unification Revisited. J. Minker, 1987.

[MA95] Y. Mulouchi and S. Arikawa. Towards a mathematical theory of machine discovery from facts. *Theoretical computer science*, 137(1):53–84, 1995.

[Paw91] Z. Pawlak. *Rough Sets.* Kluwer Academic, Dordrecht, 1991.

[Plo70] G.D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5, 1970.

[Pre97] Susanne Prediger. Logical scaling in formal concept analysis. In D. Lukose, H. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual structures: Fulfilling Peirce's dream*, number 1257 in Lecture Notes in Artificial Intelligence, Berlin–Heidelberg–New York, 1997. Springer–Verlag.

[SP98] A. Skowron and L. Polkowski. *Rough Sets in Knowledge Discovery.* Physica-Verlag, Heidelberg, 1998.

[Stu95] G. Stumme. Exploration tools in formal concept analysis. Technical report, Technische Hochschule Darmstadt, Shlossgartenst.7, D-64289 Darmstadt, Germany, 1995.

[Tc89] A Thayse and col. *Approche logique de l'Intelligence artificielle.* Dunod, Paris, 1989. in French.

[TC96] C. Tessier and L. Chaudron. Constructive difference and disagreement: a SuprA-cooperation among agents. *CSCW: The journal of collaborative Computing*, 5:323–336, 1996.

[Ver75] S. Vere. Induction of concepts in the predicate calculus. In *IJCAI'75 Proceedings of the 4th Intl. Conf. on Artificial Intelligence*, pages 281–287, Tbilissi, USSR, 3-8 september 1975.

[Zic91] M. Zickwolf. *Rule exploration: First Order Logic in Formal Concept Analysis.* PhD thesis, THD Darmstadt University, 1991. Short English version, nr1580, June 1993.