# Arbitrary Relations in Formal Concept Analysis and Logical Information Systems

Sébastien Ferré, Olivier Ridoux, and Benjamin Sigonneau

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

**Abstract** A logical view of formal concept analysis considers attributes of a formal context as unary predicates. In a first part, we propose an augmented definition that handles *binary relations* between objects. A Galois connection is defined on augmented contexts. It represents concept inheritance as usual, but also relations between concepts. As usual, labeling operators are also defined. In particular, concepts and relations are visible and labeled in a single structure. In a second part, we show how relations can be used for navigating in an augmented concept lattice. This part augments the theory of Logical Information Systems. An implementation is sketched, and first experimental results are presented.

## 1 Motivation

Previous works have shown how FCA can serve as a basis for navigating in a set of objects [1,2,3], automated learning [4], and other operations like querying, datamining, and context updating [5]. All these works show the versatility of FCA and of its basic schema of a Galois connection between sets of objects and their descriptions. The Galois connection yields a concept lattice structure that is the formal foundation for navigating, infering, approximating, etc. However in most applications of FCA, and Logical Information Systems (LIS) [5] in particular, objects are described in isolation; no explicit relations between objects can be described. Still, many applications are better modelled by arbitrary relations between objects rather than by atomic objects only: e.g., in software engineering and geographical information systems.

Power context families [6] introduce arbitrary relations in formal contexts but not in the concept lattice of objects, which is used as a basis for navigating, querying, and datamining. So, we want to extend the definitions of Galois connection, intent, concept lattice, and labeling to a power context family as a whole, instead to each context of the family in isolation, while focusing on concepts whose extent is a set of objects. Another way to say it is that we want to incorporate relations in the description of objects, and relation concept lattices in the object concept lattice. A direct application will be to augment navigation in LIS by allowing to follow relations of the formal context between concepts in addition to hierarchical relations.

Though this work applies equally well to standard FCA, we express it in terms of logical concept analysis [7] because intentions of relations will take the form of quantified formulas, which fits better a framework in which logic formulas are native.

The structure of the sequel is as follows. Section 2 presents other attempts to introduce relations in FCA, and logical formalisms that have inspired this work. Section 3 describes our proposal. Section 4 presents its application to navigation in a logical information system, and Section 5 sketches its implementation as a file system, and an example.

## 2 Related Work

The main attempt to incorporate relations in FCA is the *power context family* [6]. It consists of a vector of formal contexts $(K_1, \ldots, K_n)$ $(n \geq 2)$ with $K_i = (\mathcal{O}_i, \mathcal{A}_i, I_i)$ $(i = 1, \ldots, n)$ such that $\mathcal{O}_i \subseteq (\mathcal{O}_1)^i$. It encompasses well arbitrary relations at the context level, including $n$-ary relations. However, a different concept lattice is generated for each context. On the contrary, we seek to define a single concept lattice, where the concepts combine information about objects, the relations they have, the objects accessible from these relations, and so on. This is because we use concept lattices as a navigation structure in order to retrieve sets of objects depending on their properties (including relations).

The handling of arbitrary relations in conceptual graphs [8] and description logics [9] is just natural, since this is precisely what they are built for. However in both formalisms, concepts are given *a priori*, and not generated from a formal context. A conceptual graph and an ABox (set of objects and relations labeled by logical properties) can be used to build a power context family, but both formalisms lack the ability to exhibit collections of objects as FCA do.

Conceptual graphs have been combined with FCA by defining the concept and relation types as the concepts of a power context family [10].

Description logics are languages of unary predicates whose most relevant feature for this article is that they handle relations (called *roles* in the realm of description logics) via quantifications. The theory of description logic tells how to express classes and test whether a class entails another or whether an individual belongs to a class. Prediger and Stumme [11] have used description logic to defined a logical scale, and build a formal context over a relational database, but this context contains no relation. Baader et al. has done some work combining FCA and description logics [12]. Besides the fact they have different objectives than ours, a key difference is that they use relations inside the representation of each object (description logic concepts), whereas we here consider explicit relations between objects of a context. In the terms of description logics, their context is a TBox (set of terminological definitions), whereas our context is an ABox.

In this paper we adopt the choice of description logics to consider only binary relations, knowing this is not really a restriction as $n$-ary relations can always be converted to binary relations through reification.

## 3 Relations in Logical Concept Analysis

### 3.1 Adding Relations to the Formal Context

Firstly, we define an *object context* that is identical to logical contexts previously defined in LCA [7].

**Definition 1 (object context).**
*An* object context *is a triple* $K_1 = (\mathcal{O}, \mathcal{L}_1, d_1)$*, where:*

- $\mathcal{O}$ *is a finite set of objects,*
- $\mathcal{L}_1 = (L_1, \sqcup_1, \perp_1)$ *is a 0-sup-semilattice.* $\mathcal{L}_1$ *can be tought as a logic : elements of* $L_1$ *are called formulas,* $\sqcup_1$ *is called disjunction,* $\perp_1$ *is the neutral element for disjunction (i.e., false) and the order* $\sqsubseteq_1$*, given by* $x \sqsubseteq_1 y \Leftrightarrow x \sqcup_1 y = y$*, is called subsumption ordering.*
- $d_1 \in \mathcal{O} \to L_1$ *is a mapping from objects to their logical* description.

**Lemma 1.** *Let* $K_1$ *be an object context. The pair of mappings* $(ext_1, int_1)$*, defined by*

$$\begin{aligned} ext_1(f_1) &= \{o \in \mathcal{O} \mid d_1(o) \sqsubseteq_1 f_1\} \quad \text{for } f_1 \in L_1 \\ int_1(O) &= \bigsqcup\nolimits_1 \{d_1(o) \mid o \in O\} \quad \text{for } O \subseteq \mathcal{O} \end{aligned}$$

*is a Galois connection between* $\mathcal{P}(\mathcal{O})$ *and* $L_1$*:* $O \subseteq ext_1(f_1) \Leftrightarrow int_1(O) \sqsubseteq_1 f_1$.

*Note 1.* $\bigsqcup_1$ is well-defined as $\mathcal{O}$ is finite.

Remark that $int_1(\emptyset)$ is $\perp_1$. Secondly, we define a *relation context* that logically describes binary relations between objects of an object context.

**Definition 2 (relation context).** *Let* $\mathcal{O}$ *be a set of objects, as in Definition 1. A* relation context *is a triple* $(\mathcal{R}, \mathcal{L}_2, d_2)$*, where:*

- $\mathcal{R}$ *is a binary relation, i.e. a set of pairs in* $\mathcal{O} \times \mathcal{O}$*, equiped with two mappings start and end, s.t.* $start((o, o')) = o$ *and* $end((o, o')) = o'$*. Moreover,* $\mathcal{R}$ *is closed w.r.t. an* inverse *operation* $^{-1}$ *such that for every relation* $r \in \mathcal{R}$*,* $start(r^{-1}) = end(r)$ *and* $end(r^{-1}) = start(r)$*.*
- $\mathcal{L}_2 = (L_2, \sqcup_2, \perp_2, .^{-1})$ *is a 0-sup-semilattice.* $\mathcal{L}_2$ *can be thougth as a logic: elements of* $L_2$ *are formulas,* $\sqcup_2$ *is called disjunction,* $\perp_2$ *is the neutral element for disjunction (i.e., false) and the order* $\sqsubseteq_2$*, given by* $x \sqsubseteq_2 y \Leftrightarrow x \sqcup_2 y = y$ *is called subsumption ordering. It must support an inverse operation* ($^{-1}$) *over formulas reflecting the inverse of relations (see below), and such that for every* $f_2, g_2 \in L_2$*, the following axioms are satisfied:*
  - $(f_2^{-1})^{-1} \equiv_2 f_2$ *(where* $f \equiv_2 g \;\; =_{\text{def}} \;\; f \sqsubseteq_2 g \wedge g \sqsubseteq_2 f$*)*
  - $f_2 \sqsubseteq_2 g_2 \Leftrightarrow f_2^{-1} \sqsubseteq_2 g_2^{-1}$ *(*$\sqsubseteq_2$ *is invariant to reading direction).*
- $d_2 \in \mathcal{R} \to L_2$ *is a mapping from binary relations to logical formulas that is compatible with the inverse operation on relations, i.e.,* $d_2(r^{-1}) \equiv_2 d_2(r)^{-1}$*, for all* $r \in \mathcal{R}$*.*

|  | male | female | dead | old | grown-up | young | 30s | 40s | 90s | none | Carl Barks | Don Rosa | Taliaferro | Osborne | Walt Disney |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| downy |  | × | × |  |  |  |  |  | × |  |  | × |  |  |  |
| fergus | × |  | × |  |  |  |  |  | × |  | × | × |  |  |  |
| matilda |  | × | × |  |  |  |  |  | × |  | × | × |  |  |  |
| scrooge | × |  |  | × |  |  |  | × |  |  | × |  |  |  |  |
| hortense |  | × | × |  |  |  |  |  | × |  | × | × |  |  |  |
| quackmore | × |  | × |  |  |  |  |  | × |  | × | × |  |  |  |
| unknown | × |  |  |  |  |  |  |  |  | × |  |  |  |  |  |
| della |  | × |  |  | × |  |  |  |  |  | × | × | × | × |  |
| donald | × |  |  |  | × |  |  |  |  |  |  |  |  |  | × |
| huey | × |  |  |  |  | × | × |  |  |  |  |  | × | × |  |
| dewey | × |  |  |  |  | × | × |  |  |  |  |  | × | × |  |
| louie | × |  |  |  |  | × | × |  |  |  |  |  | × | × |  |

|  | husband | mother |
|---|---|---|
| (downy, fergus) | × |  |
| (hortense, quackmore) | × |  |
| (della, unknown) | × |  |
| (hortense, downy) |  | × |
| (scrooge, downy) |  | × |
| (mathilda, downy) |  | × |
| (donald, hortense) |  | × |
| (della, hortense) |  | × |
| (huey, della) |  | × |
| (dewey, della) |  | × |
| (louie, della) |  | × |

**Figure1.** Object and relation context for the Duck's family example.

*Example 1.* Figure 1 represents the object and relation context for the members of Donald Duck's family. This example takes place in classical FCA, which is a special case of LCA where $L_1 = \mathcal{P}(A_1), \bot_1 = \emptyset$ and $\sqcup_1 = \cap$, so that $d_1(\text{downy}) = \{\text{female}, \text{dead}, 90\text{s}, \text{Don Rosa}\}$. $\mathcal{L}_2$ is defined in a similar fashion. Note that according to Definition 2 the second table should actually be closed by $^{-1}$ to hold a relation context.

**Lemma 2.** *Let $K_2$ be a relation context. The pair of mappings $(ext_2, int_2)$, defined by*

$$\begin{aligned} ext_2(f_2) &= \{r \in \mathcal{R} \mid d_2(r) \sqsubseteq_2 f_2\} \quad \textit{for } f_2 \in L_2 \\ int_2(R) &= \bigsqcup\nolimits_2 \{d_2(r) \mid r \in R\} \quad \textit{for } R \subseteq \mathcal{R} \end{aligned}$$

*is a Galois connection between $\mathcal{P}(\mathcal{R})$ and $L_2$: $R \subseteq ext_2(f_2) \Leftrightarrow int_2(R) \sqsubseteq_2 f_2$.*

*Note 2.* $\bigsqcup_2$ is well-defined as $\mathcal{O}$ is finite.

The idea is now to combine both contexts in a context $K$, and both logics in a logic $\mathcal{L}$ similar to description logics in that relations can be used to retrieve objects depending on their relationships to other objects.

**Definition 3 (combined context and logic).** *Let $K_1$ be an object context, and $K_2$ be a relation context. The* combined context *is the pair $(K_1, K_2)$ gathering objects, relations, and their logical descriptions. The* combined logic $\mathcal{L}$ *is the couple $(L, \sqsubseteq)$, where:*

– $L \longrightarrow \top \mid L_1 \mid \exists L_2.L$

– $f \sqsubseteq g \Leftrightarrow \begin{cases} \mathit{true} & \mathit{if}\ g = \top \\ f \sqsubseteq_1 g & \mathit{if}\ f, g \in L_1 \\ (f_2 \sqsubseteq_2 g_2) \wedge (f' \sqsubseteq g') & \mathit{if}\ f = \exists f_2.f', g = \exists g_2.g'\ (\mathit{where}\ f', g' \in L) \\ \mathit{false} & \mathit{otherwise} \end{cases}$

*Note 3.* Formulas of L can be expressed in first-order predicate logic in the following manner. $L_1$ and $L$ are sets of unary predicates, and $L_2$ is a set of binary predicates such that $\exists f_2.f$ is defined by $(\exists f_2.f)(x) \Leftrightarrow \exists x'.(f_2(x, x') \wedge f(x))$.

*Note 4.* For convenience, we extend the subsumption $\sqsubseteq$ to sets of formulas $G, F \subseteq L$ by defining $G \sqsubseteq f \iff \exists g \in G : g \sqsubseteq f$, and $G \sqsubseteq F \iff \forall f \in F : G \sqsubseteq f$.

We now prove there exists a Galois connection between sets of objects and sets of formulas from $\mathcal{L}$, and we give a definition of it as a couple $(\mathit{ext}, \mathit{int})$.

**Definition 4 (extent).** *Let $K$ and $\mathcal{L}$ be the context and logic combined from $K_1$ and $K_2$. The* extent *in $K$ of a set of formulas $F \subseteq L$ is a set of objects defined by $\mathit{ext}(F) = \bigcap_{f \in F} \mathit{ext}'(f)$, where*

$$\mathit{ext}'(f) = \begin{cases} \mathcal{O} & \mathit{if}\ f = \top \\ \mathit{ext}_1(f) & \mathit{if}\ f \in L_1 \\ \{\mathit{start}(r) \mid r \in \mathit{ext}_2(f_2), \mathit{end}(r) \in \mathit{ext}'(f')\} & \mathit{if}\ f = \exists f_2.f'. \end{cases}$$

**Definition 5 (intent).** *Let $K$ and $\mathcal{L}$ be the context and logic combined from $K_1$ and $K_2$. The* intent *in $K$ of a set of objects $O \subseteq \mathcal{O}$ is defined by*

$$\mathit{int}(O) = \{f \in L \mid O \subseteq \mathit{ext}'(f)\}.$$

**Theorem 1 (relational Galois connection).** *Given a combined context $K$, the pair of mappings $(\mathit{ext}, \mathit{int})$ defined above is a Galois connection between $(\mathcal{P}(\mathcal{O}), \subseteq)$ and $(\mathcal{P}(L), \supseteq)$, i.e. for every $O \subseteq \mathcal{O}$ and $F \subseteq \mathcal{L}$:*
$$O \subseteq \mathit{ext}(F) \Leftrightarrow \mathit{int}(O) \supseteq F.$$

Then it is well known from FCA [13] that a complete concept lattice can be defined.

**Theorem 2 (concept lattice).** *Let $K$ be a combined context. Let the set of* concepts *$\mathcal{C}$ be defined as the set of all pairs $(O, F) \in \mathcal{P}(\mathcal{O}) \times \mathcal{P}(L)$ such that $O = \mathit{ext}(F)$ and $F = \mathit{int}(O)$: $O$ is called the* extent*, and $F$ is called the* intent*. The partial ordering $(\mathcal{C}, \leq)$, where $(O, F) \leq (O', F') \iff O \subseteq O'$, is a complete lattice, the* concept lattice*, as a direct consequence of Theorem 1.*

It is important here to note that concept intents gather properties about objects as formulas in $\mathcal{L}_1$, properties about existing relations as formulas like $\exists f_2.\top$, and recursively properties about related objects as formulas like $\exists f_2.f'$.

Now, a problem with Definition 5 is that it suggests that intents can be computed only by testing every formula in $L$. In the following we show that approximations of these intents can be effectively computed, and this to an arbitrary accuracy.

Firstly, we define $L(n)$ as the subset of $L$ containing every formula that have no more than $n$ times the existential quantifier $\exists$. This corresponds to restricting the depth of relation paths to $n$.

**Definition 6 (depth-$n$ intent).** *Let $K$ and $\mathcal{L}$ be the context and logic combined from $K_1$ and $K_2$. The* depth-$n$ intent *$int(n)(O)$ in $K$ of a set of objects $O \subseteq \mathcal{O}$ is defined by*

$$\begin{aligned} int(0)(O) &= int_1(O) \\ int(n+1)(O) &= int(n)(O) \\ &\cup \{\exists int_2(R).f' \mid \exists R \subseteq \mathcal{R}, O = start(R), int(n)(end(R)) \sqsubseteq f'\}. \end{aligned}$$

This definition is well-founded, and because of the finiteness of objects and relations every approximate intent is also finite. However they can represent an infinite set of formulas thanks to subsumption $\sqsubseteq$ in $L$: if $f \notin int(n)(O)$ but $int(n)(O) \sqsubseteq f$, then $f$ implicitly belongs to the intent of $O$ at depth $n$.

**Theorem 3 (depth-$n$ relational Galois connection).** *Given a combined context $K$ and a depth $n$, the pair of mappings $(ext, int(n))$ is a Galois connection between $(\mathcal{P}(\mathcal{O}), \subseteq)$ and $(\mathcal{P}(L(n)), \sqsubseteq)$, i.e. for every $O \subseteq \mathcal{O}$ and $F \subseteq L(n)$, $O \subseteq ext(F) \Leftrightarrow int(n)(O) \sqsubseteq F$.*

*Proof.* We split proof in three parts.

1. Firstly, we prove that for all $n \in \mathbb{N}$, $O \subseteq \mathcal{O}$ and $f \in L(n)$, $O \subseteq ext'(f) \Rightarrow int(n)(O) \sqsubseteq f$. The proof works by recurrence on the depth $n$, and by induction on the syntax of formulas. The case where $n = 0$ follows from Lemma 1, so we only show the general case $n+1$, when $f = \exists f_2.f'$, i.e., $f \in L(n+1)$, and so $f' \in L(n)$.
$O \subseteq ext'(\exists f_2.f') \Longrightarrow O \subseteq \{start(r) \mid r \in ext_2(f_2), end(r) \in ext'(f')\}$
$\Longrightarrow \forall o \in O : \exists r \in ext_2(f_2) : o = start(r), end(r) \in ext'(f')$
$\Longrightarrow \exists R \subseteq ext_2(f_2) : O = start(R), end(R) \subseteq ext'(f')$
$\Longrightarrow \exists R \subseteq \mathcal{R} : O = start(R), R \subseteq ext_2(f_2), end(R) \subseteq ext'(f')$
$\Longrightarrow \exists R \subseteq \mathcal{R} : O = start(R), int_2(R) \sqsubseteq_2 f_2, int(n)(end(R)) \sqsubseteq f'$
(Lemma 2, recurrence hypothesis because $f' \in L(n)$)
$\Longrightarrow \exists R \subseteq \mathcal{R} : O = start(R), int(n)(end(R)) \sqsubseteq f', \exists int_2(R).f' \sqsubseteq \exists f_2.f'$
$\Longrightarrow \exists g \in int(n+1)(O) : g \sqsubseteq f$ $\quad (g = \exists int_2(R).f')$
$\Longrightarrow int(n+1)(O) \sqsubseteq f$.
2. Secondly, we prove in the same way the reciprocal lemma, i.e. $int(n)(O) \sqsubseteq f \Rightarrow O \subseteq ext'(f)$.
Suppose $int(n+1)(O) \sqsubseteq \exists f_2.f'$
Either $int(n)(O) \sqsubseteq \exists f_2.f' \Longrightarrow O \subseteq ext'(\exists f_2.f')$ (recurrence hypothesis)
or $\{\exists int_2(R).f' \mid R \subseteq \mathcal{R}, O = start(R), int(n)(end(R)) \sqsubseteq f'\} \sqsubseteq \exists f_2.f'$
$\Longrightarrow \exists R \subseteq \mathcal{R} : O = start(R), int(n)(end(R)) \sqsubseteq f', \exists int_2(R).f' \sqsubseteq \exists f_2.f'$
$\Longrightarrow \exists R \subseteq \mathcal{R} : O = start(R), int_2(R) \sqsubseteq_2 f_2, int(end(R)) \sqsubseteq f'$
$\Longrightarrow \exists R \subseteq \mathcal{R} : O = start(R), R \subseteq ext_2(f_2), end(R) \subseteq ext'(f')$

(Lemma 2, recurrence hypothesis because $f' \in L(n)$)
$\Longrightarrow \forall o \in O : \exists r \in \mathit{ext}_2(f_2) : o = \mathit{start}(r), \mathit{end}(r) \in \mathit{ext}'(f')$
$\Longrightarrow O \subseteq \mathit{ext}'(\exists f_2.f') \Longrightarrow O \subseteq \mathit{ext}'(f)$.

3. Finally, we prove the theorem for all $n \in \mathbb{N}$, $O \subseteq \mathcal{O}$ and $f \in L(n)$.
$O \subseteq \mathit{ext}(F) \iff \forall f \in F : O \subseteq \mathit{ext}'(f) \iff \forall f \in F : \mathit{int}(n)(O) \sqsubseteq f$ (first and second part of this proof) $\iff \mathit{int}(n)(O) \sqsubseteq F$. □

This last result entails that for every depth $n$ a Galois connection is defined, and so, a depth-$n$ concept lattice can be derived from it. We complete this by showing that when the depth tends to infinity the depth-$n$ intent is equivalent to the *full* intent of Definition 5.

**Theorem 4 (limit relational Galois connection).** *Let $K$ be a combined context. For every set of objects $O$, when the depth $n$ tends to infinity, the set of formulas in $L$ that are subsumed by the depth-$n$ intent tends to be equal to the full intent $\mathit{int}(O)$, i.e.*

$$\forall O \subseteq \mathcal{O} : \forall f \in L : f \in \mathit{int}(O) \iff \exists n \in \mathbb{N} : \mathit{int}(n)(O) \sqsubseteq f.$$

*Proof.* $f \in \mathit{int}(O) \iff O \subseteq \mathit{ext}'(f) \iff O \subseteq \mathit{ext}(\{f\})$
$\iff \exists n \in \mathbb{N} : \mathit{int}(n)(O) \sqsubseteq \{f\}$ (Theorem 3 because $f \in L(n)$)
$\iff \exists n \in \mathbb{N} : \mathit{int}(n)(O) \sqsubseteq f$. □

The exact full intents and concept lattice cannot be computed, especially if there is a cycle between objects related by $\exists f_2.f'$ formulas. But this is not really a problem since Theorems 3 and 4 show that we have a series of finite depth-$n$ intents and related concept lattices, which can be made as close as possible to *full* intents and concept lattice.

### 3.2 Adding Relations to the Concept Lattice Labeling

All the information contained in the binary relation context is present in the concept lattice $\mathcal{C}$, and can be made explicit by adding a relational labeling to the concept lattice, in addition to the usual labeling by objects and formulas.

**Definition 7 (labeling).** *Let $\mathcal{C}$ be a concept lattice. The labeling of $\mathcal{C}$ by formulas, noted $\mu$, and by objects, noted $\gamma$, are defined as follows:*
$\mu \in L \to \mathcal{C}, \quad \mu(f) = (\mathit{ext}(\{f\}), \mathit{int}(\mathit{ext}(\{f\})))$,
$\gamma \in \mathcal{O} \to \mathcal{C}, \quad \gamma(o) = (\mathit{ext}(\mathit{int}(\{o\})), \mathit{int}(\{o\}))$.

It is well-known that a concept lattice contains the same information as the object context from which it derives. We show now that the concept lattice derived from a combined context also contains its relational information.

A way of showing that $\mathcal{C}$ contains in some way the relation concept lattice $\mathcal{C}_2$ is to build an order-preserving mapping from the latter to pairs of concepts of the former. So we go on defining a squared version of $\mu$, and $\gamma$, applying to pairs of concepts.

**Definition 8 (relational labeling).** *Let $c, c' \in \mathcal{C}$. We define:*
$\mu^2 \in \mathcal{L}_2 \to \mathcal{C}^2, \mu^2(f_2) = (\mu(\exists f_2.\top), \mu(\exists f_2^{-1}.\top))$,
$\gamma^2 \in \mathcal{R} \to \mathcal{C}^2, \gamma^2(r) = (\gamma(start(r)), \gamma(end(r)))$.

This implies that in addition to subsumption links between concepts ($\leq$), there are relation links between concepts (either individual relations between object-labeled concepts, or relational formulas between formula-labeled concepts). Moreover, most properties on labeling functions are kept [13,5]. In the following, the ordering on pairs of concepts is defined by: $(c_1, c_1') \leq (c_2, c_2')$ iff $c_1 \leq c_2$ and $c_1' \leq c_2'$; and the inverse of a pair of concept is defined by: $(c, c')^{-1} = (c', c)$.

There exists an order-preserving mapping from the concept lattice of the relation context $K_2$ into the concept lattice of the combined context $K = (K_1, K_2)$.

**Theorem 5 (order-preserving mapping).** *The mapping $\varphi \in \mathcal{C}_2 \to \mathcal{C}^2$, $\varphi(c_2) = \mu^2(int_2(c_2))$ is order-preserving, i.e:*

$$\forall c_2, c_2' \in \mathcal{C}_2 : c_2 \leq c_2' \Rightarrow \varphi(c_2) \leq \varphi(c_2').$$

**Corollary 1.** *A corollary of Theorem 5 is that the ordering between formula-labels is preserved: $\forall f_2, f_2' \in L_2$, $\mu_2(f_2) \leq \mu_2(f_2') \Rightarrow \mu^2(f_2) \leq \mu^2(f_2')$.*

So, each time two relation labels are ordered in the relation concept lattice $\mathcal{C}_2$, they are so in the combined concept lattice $\mathcal{C}$. However the reverse does not hold. For example, consider the two relations `parent` and `grand-parent`, whose inverse are respectively `child` and `grand-child`. It is true that anyone who has a grand-parent also has a parent ($\mu(\exists \textit{grand-parent}.\top) \leq \mu(\exists \textit{parent}.\top)$); reciprocally, anyone who has a grand-child also has a child ($\mu(\exists \textit{grand-child}.\top) \leq \mu(\exists \textit{child}.\top)$). This implies the label `grand-parent/grand-child` is lower than the label `parent/child` in the combined concept lattice. But this is certainly not the case in the relation concept lattice as a relation can never have both properties `parent/child` and `grand-parent/grand-child`. In conclusion the combined concept lattice can add useful implications compared to the relation concept lattice.

*Example 2.* Figure 2 shows the depth-1 concept lattice built from the combined context of Example 1, and its labeling by attributes, objects, and relational properties $\exists r.\top$. Grey circles represent the concepts introduced by relations, dashed arrows and italic labels stand for labeling by relations.

It shows that every mother has a husband, and reciprocally that every married female is a mother. Hence, every duck who has a mother also has a father.

## 4 Querying and Navigating with Relations in LIS

In previous work about Logical Information Systems (LIS) [5], querying and navigation are defined on an object context $K_1 = (\mathcal{O}, \mathcal{L}_1, d_1)$. Queries are formulas in $\mathcal{L}_1$, and for every query $q \in \mathcal{L}_1$, $ext_1(q)$ is the set of answers to the

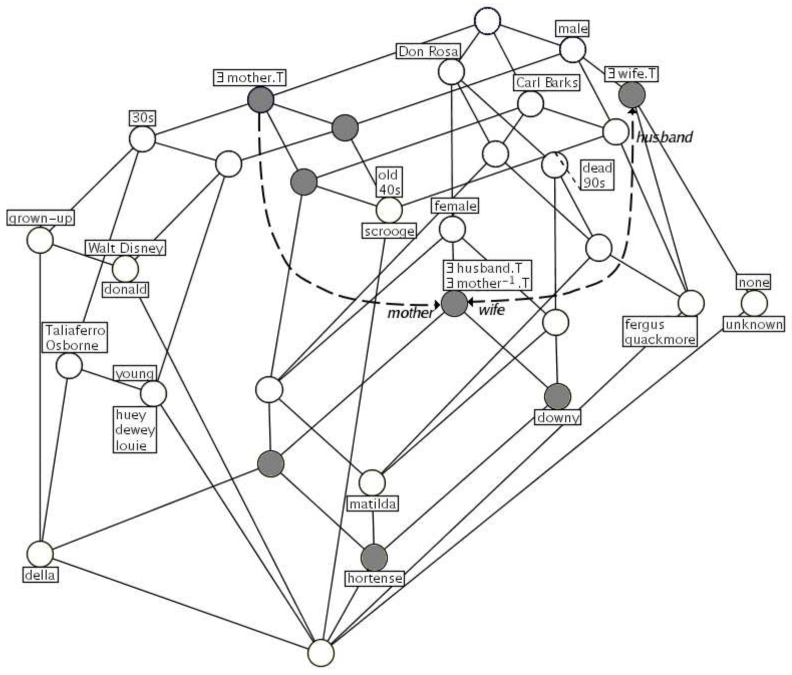

**Figure2.** Concept lattice labeled by formulas, objects and some relations

query. In order to help users building queries, even without knowledge about both the context and the logic, a set of navigation links can be computed for any query $q$ in order to refine it. Navigation links are not searched in the whole space of logical formulas. They are searched in a subset of $L_1$ which we call navigation features, $feat_1(K_1)$, that depends on the context and is user-defined. The smaller and simpler formulas are in this subset, the simpler and more efficient the navigation is.

The set of navigation links $dirs(q)$ that can refine a query $q$ is defined by:

$$dirs(q) = Max_{\sqsubseteq_1}\{x \in feat_1(K_1) \mid \emptyset \subsetneq ext_1(q \sqcap_1 x) \subsetneq ext_1(q)\}.$$

In order to have an efficient and progressive navigation, links computed by $dirs(q)$ must carry information, $ext_1(q \sqcap_1 x) \subsetneq ext_1(q)$, not lead to a dead-end, $\emptyset \subsetneq ext_1(q \sqcap_1 x)$, and be as general as possible w.r.t. subsumption ordering: $Max_{\sqsubseteq_1}$.

Each link $x$ enables the user to move from the query $q$ to the query $q \sqcap_1 x$. Finally individual objects have to be found in some place. Hence the definition of local objects for any query $q$:

$$locals(q) = ext_1(q) \setminus \bigcup_{x \in dirs(q)} ext_1(x).$$

Note that the definition of *dirs* relies on a conjunction operation $\sqcap_1$ on queries. However, such a connective was not required on the development of logical concept analysis. Similarly, navigation is supposed to start from the top element of the logic, *true* or $\top_1$, whereas it is the bottom element, *false* or $\bot_1$ that is required in logical concept analysis. This is not a contradiction, but this must be examined in the actual definition of a querying and navigation system.

In the following we present the relational extension of querying and navigation in formal contexts.

## 4.1 Querying: Query Language and Extent

In order to have boolean operators in the *query language* $L_q$, we extend the combined logic $L$ in the following way:

$$L_q \to \top \mid \bot \mid L_q \wedge L_q \mid L_q \vee L_q \mid \neg L_q \mid L_1 \mid \exists L_2.L_q.$$

It is not necessary to extend the subsumption $\sqsubseteq$ to the query language as will be made clear later. However we do need to define the extent of queries, given some context $K = (K_1, K_2)$ in order to answer queries.

**Definition 9 (query extent).** *Let $K$ be a combined context from contexts $K_1$ and $K_2$.*

$$\begin{array}{llll} ext_q(\top) &= \mathcal{O} & ext_q(q_1 \wedge q_2) &= ext_q(q_1) \cap ext_q(q_2) \\ ext_q(\bot) &= \emptyset & ext_q(q_1 \vee q_2) &= ext_q(q_1) \cup ext_q(q_2) \\ ext_q(f_1 \in L_1) &= ext_1(f_1) & ext_q(\neg q) &= \mathcal{O} \setminus ext_q(q) \\ ext_q(\exists f_2.q) &= \{start(r) \mid r \in ext_2(f_2), end(r) \in ext_q(q)\} & & \end{array}$$

The last 2 lines are taken from Definition 4, given that in these cases, queries are in the logic $\mathcal{L}$.

## 4.2 Features: Useful Features for Navigation

In this section we show that not all formulas in $L_q$ need to be considered as candidates for navigation links. This leads to a definition of $feats_q(K)$ that is only a small subset of the full query language itself.

Firstly we define $X(q)$ as the set of possible navigation links for some query $q$, i.e., formulas satisfying the "strictly refining and relevant" property:

$$X(q) = \{x \in L_q \mid \emptyset \subsetneq ext_q(q \wedge x) \subsetneq ext_q(q)\}.$$

This definition differs from the definition of *dirs* by the fact that the selection of subsumption-maximal elements is not applied, and that the full language $L_q$ is considered instead of a subset of features $feat_q(K)$. Our purpose is precisely to characterize the latter.

**Lemma 3 (elimination of connectives).** *For every query $q \in L_q$,*

$$\begin{array}{ll} q_1 \wedge q_2 \in X(q) \Rightarrow q_1 \in X(q) \vee q_2 \in X(q), & \top \notin X(q) \\ q_1 \vee q_2 \in X(q) \Rightarrow q_1 \in X(q) \vee q_2 \in X(q), & \bot \notin X(q) \\ \neg q_1 \in X(q) \Rightarrow q_1 \in X(q) & \end{array}$$

This indicates that boolean connectors can all be ignored in features. For example, consider the proposition about conjunction, and let $q_1 \wedge q_2 \in X(q)$. Either both $q_1$ and $q_2$ are in $X(q)$, and so $q_1 \wedge q_2$ can be obtained by successively selecting $q_1$ and $q_2$; or only one subquery, say $q_1$, is in $X(q)$, which means that $q_1 \wedge q_2$ is in fact equivalent to $q_1$ for navigation (they reach the same concept).

**Lemma 4.** *For all queries $q, q' \in L_q$, and all relation formula $x_2 \in L_2$,*

1. $\exists x_2.q' \in X(q) \Leftrightarrow q' \in X(\exists x_2^{-1}.q)$,
2. $\exists x_2.q' \in X(q) \Rightarrow ext_q(q \wedge \exists x_2.\top) \neq \emptyset$.

Lemma 4 allows to recursively decompose the search for relational links. It shows this can be achieved by first looking for $\exists x_2.\top$ formulas that make the new query non-empty (second proposition), and then by replacing the $\top$ by navigation links among the images by $x_2$ of objects in $q$ (first proposition).

Hence, the set of useful navigation features for the language $L_q$ is as follows.

**Definition 10 (features).**
$feat_q(K) = feat_1(K_1) \cup \{\exists x_2.x \mid x_2 \in feat_2(K_2), x \in feat_q(K)\}$

This shows that the useful part of the query language for navigation is included in $L$, even if general queries do not belong to $L$. Feature sets of logics $L_1$ and $L_2$, $feat_1(K_1)$ and $feat_2(K_2)$ are supposed to be defined with their respective contexts. Note that if the query language in contexts $K_1$ and $K_2$ is based on propositional calculus, variants of Lemma 3 will hold; $feat_1(K_1)$ and $feat_2(K_2)$ are just the atomic formulas of the logic of their context.

### 4.3 Navigation: Links and Local Objects, Selection and Traversal

We can now define a version of *dirs* that is extended with relations.

**Definition 11 (Navigation links).**

$$dirs(q) = Max_{\sqsubseteq}\{x \in feat_q(K) \mid \emptyset \subsetneq ext_q(q \wedge x) \subsetneq ext_q(q)\}.$$

*There is no problem with using the subsumption $\sqsubseteq$ because $feat_q(K) \subseteq L$.*

Local objects are defined as usual: an object is local if it not accessible from any navigation link. Navigation links in a combined relation and object context are of two kinds: object formulas, and relation formulas. Object formulas are used in logical queries, but relation formulas can be used either in logical queries, or as paths to go through relations. In each case following a link transforms the current query $q$ as follows (assignement is denoted by :=).

1. A link $x_1 \in L_1$ is used as usual to refine a query: $q := q \wedge x_1$. For instance, property *old* can be such a refinement.
2. A link $\exists x_2.x \in L_2$ can also be used for refining a query, except that the refinement applies to objects in relation to current objects instead of the current objects themselves: $q := q \wedge \exists x_2.x$. For instance, property $\exists parent.old$ can be used to select individuals that have an old parent.

3. A link $\exists x_2.x \in L_2$ also means that relations of type $x_2$ can be traversed to reach objects of type $x$: $q := x \wedge \exists x_2^{-1}.q$. Only those objects of type $x$ that can be reached from $q$ through $x_2$ are considered. For instance, property $\exists \mathit{parent}.\mathit{old}$ can be used to select the old parents of the curently selected individuals.

Cases 1 and 2 are kinds of *conceptual navigation* (from some concept to a subconcept), whereas case 3 is a kind of *relational navigation* (from some concept to a related concept).

One may want to consider a query like $\exists x_2.(x \wedge x')$ (consider, say, $\exists \mathit{parent}.(\mathit{old} \wedge \mathit{loving})$ for old and loving parents). Lemma 3 shows that it is not necessary to produce navigation links of this form. However, if we successively select links $\exists x_2.x$ and $\exists x_2.x'$, then the resulting query is $\exists x_2.x \wedge \exists x_2.x'$, which is not equivalent to $\exists x_2.(x \wedge x')$. This is where relational navigation comes in for help: first, traverse $x_2$ ($q := \exists x_2^{-1}.\top$), then select $x$ ($q := \exists x_2^{-1}.\top \wedge x$), and select $x'$ ($q := \exists x_2^{-1}.\top \wedge x \wedge x'$), and finally, traverse $x_2^{-1}$ ($q := \exists x_2.(\exists x_2^{-1}.\top \wedge x \wedge x') \equiv \exists x_2.(x \wedge x')$).

## 5 Implementation as a File System

Though a LIS can be implemented as a stand-alone application, a parallel between LIS notions and file system notions makes it natural to implement it as a file system. This is the purpose of LISFS (LIS File System [14]) which is a file system that offers the operations of a LIS at the operating system level. This is achieved by considering files as objects, directories as formulas and paths as conjunctions of directories. The root directory (`/` under UNIX) plays the role of the $\top$ formula. Thus, the absolute name of a file stands as its logical description.

Commands have essentially the same effects as UNIX shell commands, w.r.t. this change. For instance, the shell command `cd a` changes the current query $q$ to $q \wedge a$, and the command `ls` is used to list the navigation links (computed by *dirs*) and the local objects (computed by *objects*) of the current query.

### 5.1 Adding Arbitrary Relations to LISFS

From a file-system point of view, arbitrary relations can somewhat be understood as symbolic links. As a matter of fact, having a relation $r$ between two objects $o$ and $o'$ declares that $o$ and $o'$ are linked and gives a description $r$ to the link. Regular UNIX links are just a special case where $r$ can only take up one value: the “synonym” relation.

Relations also extend the notion of symbolic link in that they can play two roles in navigation. Indeed, they can be used for a conceptual navigation, in which case they behave as normal properties, and for relational navigation. In the case of relational navigation, traversing a navigation link in LISFS is the counterpart of following a symbolic link in UNIX, with the additional benefit that links in LISFS have an inverse and apply to sets of objects.

```
[1]# ls
total 12
1       first_appearance:40s/
1       first_appearance:none/
1       old/
2       grown-up/
3       husband<true/
3       husband>true/
3       mother<true/
3       young/
4       female/
5       dead/
5       first_appearance:30s/
5       first_appearance:90s/
8       male/
8       mother>true/
11      creator:/
[2]# cd male; ls
total 8
1       first_appearance:40s/
1       first_appearance:none/
1       grown-up/
1       old/
2       dead/
2       first_appearance:90s/
3       husband<true/
3       young/
4       first_appearance:30s/
5       mother>true/
7       creator:/
[3]# cd husband<true; ls
total 3
1       first_appearance:none/
1       husband<creator:osborne/
1       husband<creator:taliaferro/
1       husband<first_appearance:30s/
1       husband<grown-up/
2       creator:/
2       dead/
2       first_appearance:90s/
2       husband<creator:carl_barks/
2       husband<dead/
2       husband<first_appearance:90s/
[4]# cd !creator: ; ls
total 1
        unknown
[5]# cat unknown
The husband of Della Duck
[6]# cd husband<< ; ls
total 1
        della
```

**Figure3.** A Running Example with the Duck Family Context

Regarding navigation and querying, the following concrete syntax is adopted ($q$ denotes the current query):

- `cd r>x` selects objects whose image by $r$ verifies $x$: $q := q \wedge \exists r.x$,
- `cd r<x` selects objects whose antecedent by $r$ verifies $x$: $q := q \wedge \exists r^{-1}.x$,
- `cd r>>` traverses relation $r$: $q := \exists r^{-1}.q$,
- `cd r<<` traverses relation $r^{-1}$: $q := \exists r.q$.

We use a slightly modified version of the shell command `ln` to create a relation between two files that takes care of the name of the relation. E.g., we write `ln r o o'` to add a relation $r$ between $o$ and $o'$.

### 5.2 Example

Figure 3 shows LISFS augmented with arbitrary relations in action. Navigation and querying take place in the Duck family context presented in Figure 1.

Starting from the $\top$ formula, command `1` asks for available navigation links. The number of objects in the current query and in its possible refinements are given. Quantified formulas appear under their concrete syntax: `mother>true` for $\exists mother.\top$, `mother<true` for $\exists mother^{-1}.\top$, etc.

From there, command `2` lets the user select the males from the members of the Duck family by setting the current query to $\top \wedge male \equiv male$. New increments are listed. Then, command `3` restricts the query to married males. Thus, increments concerning relation $husband^{-1}$ become more precise: two ducks have a wife that is dead ($\exists husband^{-1}.dead$, shown as `husband<dead`) and so on.

Moreover, three ducks in the family satisfy the current query, among whom only two of them have a creator. This seems odd, so the user asks to see ducks that does not have a creator with command `4` (`!` denotes negation). This duck is `unknown`. As objects are files in LISFS, they have a content which command `5` lists. Here, the user sees that `unknown` is supposed the be the husband of Della. To verify that this really holds in the context, the user then traverses relation $husband^{-1}$ with command `6`.

## 6 Conclusion

In this paper we have shown that relations can be smoothly introduced not only in formal contexts as is done by power context families, but also in the definition of intents, and hence in concept lattices and their labeling. The advantage over previous approaches is that information from both object and relation contexts is combined in a single concept lattice. This enables a natural but powerful extension of LIS navigation and querying [5]. Relational features express properties over objects w.r.t. their related objects, and can be used both for refining a set of objects, as usual, and for traversing some relation from a set of objects to another. This relational navigation has been implemented as an extension to an existing LIS file system (LISFS).

Although relations in contexts are arbitrary, including non tree-like structures, our query language allows only tree-like queries, as in description logics, but unlike conceptual graphs. We plan to extend the query language so as to remove this tree-like constraint. This could be done by inserting variables in queries, like in $\exists r_1.\exists r_2.X : f' \wedge \exists r_3.\exists r_4.X$ where $X$ must refer to a same object in both occurrences. We also plan to extend the logic so as to handle more complex path patterns (e.g., regular expressions), such as $\exists parent^+.famous$ meaning “has a famous ancestor”.

## References

1. Godin, R., Missaoui, R., April, A.: Experimental comparison of navigation in a galois lattice with conventional information retrieval methods. International Journal of Man-Machine Studies **38** (1993) 747–767
2. Lindig, C.: Concept. In Köhler, J., Giunchiglia, F., Green, C., Walther, C., eds.: IJCAI95 Workshop on Formal Approaches to the Reuse of Plans, Proofs, and Programs, Montreal, Canada (1995)
3. Ferré, S., Ridoux, O.: A file system based on concept analysis. In Savig, Y., ed.: Int. Conf. Rules and Objects in Databases. Volume 1861 of LNCS., Imperial College, London, UK, Springer (2000) 1033–1047
4. Ganter, B., Kuznetsov, S.: Formalizing hypotheses with concepts. In Mineau, G., Ganter, B., eds.: Int. Conf. Conceptual Structures. Number 1867 in Lecture Notes in Computer Science, Darmstadt, Germany, Springer (2000) 342–356
5. Ferré, S., Ridoux, O.: An introduction to logical information systems. Information Processing & Management **40** (2004) 383–419

6. Wille, R.: Conceptual graphs and formal concept analysis. In: Int. Conf. Conceptual Structures. Volume 1257 of LNCS., Seattle, Washington, USA, Springer (1997) 290–303
7. Ferré, S., Ridoux, O.: A logical generalization of formal concept analysis. In Mineau, G., Ganter, B., eds.: Int. Conf. Conceptual Structures. Number 1867 in Lecture Notes in Computer Science, Darmstadt, Germany, Springer (2000) 371–384
8. Sowa, J.F.: Conceptual structures. Information processing in man and machine. Addison-Wesley, Reading, MA, USA (1984)
9. Brachman, R.J.: On the epistemological status of semantic nets. In Findler, N.V., ed.: Associative Networks: Representation of Knowledge and Use of Knowledge by Examples. Academic Press, New York, USA (1979)
10. Mineau, G., Stumme, G., Wille, R.: Conceptual structures represented by conceptual graphs and formal concept analysis. In Tepfenhart, W.M., Cyre, W.R., eds.: Int. Conf. Conceptual Structures. Volume 1640 of LNCS., Blacksburg, Virginia, USA, Springer (1999) 423–441
11. Prediger, S., Stumme, G.: Theory-driven logical scaling. In: International Workshop on Description Logics. Volume 22., Sweden (1999)
12. Baader, F., Sertkaya, B.: Applying formal concept analysis to description logics. In Eklund, P.W., ed.: Int. Conf. Formal Concept Analysis. Volume 2961 of LNCS., Sydney, Australia, Springer (2004) 261–286
13. Ganter, B., Wille, R.: Formal Concept Analysis — Mathematical Foundations. Springer (1999)
14. Padioleau, Y., Ridoux, O.: A logic file system. In: USENIX Annual Technical Conference, General Track, San Antonio, Texas, USA, USENIX (2003) 99–112