Realization Problem for Formal Concept Analysis[⋆]

N.V. Shilov

Lavren'ev av., 6, Novosibirsk 630090, Russia
shilov@iis.nsk.su

Abstract. Formal Concept Analysis (FCA) is an approach to knowledge engineering, representation, and analysis. A 'standard' FCA-workflow starts with some 'experimental' data, classifies "objects" and their "attributes" in the data, represents relations between objects and attributes by a number of cross-tables (matrices), couples compatible sets of objects and attributes into concepts, and builds a number of concept lattices. In contrast, FCA realization problem starts with 'fragments' of some cross-tables and concept lattices and try to generate data that match these fragments of cross-tables and concept lattices simultaneously. The paper defines formally the FCA realization problem and proves that it is decidable. It is done by reduction of the problem to the consistency problem for description logic $\mathcal{ALBO}$. But it is still an open question whether FCA-realization problem have better complexity bound than $\mathcal{ALBO}$ has.

1 Introduction

Formal Concept Analysis (FCA) by R. Wille and B. Ganter [2] is an algebraic framework for knowledge engineering, representation, and analysis. It takes some 'experimental' data, classifies "objects" and their "attributes" in the data. Then mathematical part of FCA takes matrices that specify relation between objects and attributes as an input and finds corresponding 'natural clusters' (sets) of attributes and 'natural clusters' (sets) of objects. Every pair of corresponding 'natural' sets of objects and attributes forms a formal concept[1]. The set of formal concepts obeys the mathematical axioms defining a lattice, and is called a concept lattice.

Let us illustrate basic FCA notions by the following example. Assume that some Computer Science Department has 1 Full Professor (F), 1 Associate Professor (A), 1 Lecture (L), 2 Tutors (T_1 and T_2), and 1 Research Programmer (R). Courses provided by the department are Programming Languages (PL), Operating Systems (OS), Data Bases (DB), and Formal Methods (FM). Research range the same topics. Sample data about staff teaching and research profile are presented in the tables 1.

⋆ This research is supported in parts by (a) by joint grant RFBR 05-01-04003-a - DFG project COMO, GZ: 436 RUS 113/829/0-1, (b) by Integration Grant n.14 Siberia Branch, Russian Academy of Science.

[1] For a distinction with Description Logic, we use a combined term 'formal concept' in the FCA context.

Teaching	PL	OS	DB	FM	AI
F	x			x	x
A	x	x	x		
L		x	x		x
T_1	x		x		x
T_2		x		x	x

Research	PL	OS	DB	FM	AI
F				x	x
A	x	x			
L		x	x		
P					x

Table 1. Sample stuff Teaching and Research profile

Both tables are matrices or *cross-tables*. These matrices establish binary relations between corresponding *objects* and *attributes*. Objects are teaching and research staff members: $Obj_{\text{Teaching}} = \{$F, A, L, T_1, $T_2\}$, $Obj_{\text{Research}} = \{F, A, L, R\}$. Attributes are teaching and research topics: $Atr_{\text{Teaching}} = Atr_{\text{Research}} = \{PL, OS, DB, FM, AI\}$. Every triple that consists of objects, attributes and a cross-table is called *formal context*. For example, ($\{$F, A, L, T_1,$T_2\}$, $\{$PL, OS, DB, FM, AI$\}$, Teaching) as well as ($\{$F, A, L, P$\}$, $\{$PL, OS, DB, FM, AI$\}$, Research) are formal contexts (that we refer by names of their matrices).

In Teaching formal context two objects A and L share two common attributes OS and DB. Moreover, there is no other shared attribute for these objects, but OS and DB only. In FCA terms this connection between a set of objects $\{$A, L$\}$ and a set of attributes $\{$OS, DB$\}$ is written in the form $\{$A, L$\}^{\uparrow\text{Teaching}} = \{OS, DB\}$. On the opposite side, A and L are the only objects that share attributes OS and DB. In FCA terms this connection between a set of attributes $\{$OS, DB$\}$ and a set of objects $\{$A, L$\}$ is written in the form $\{$OS, DB$\}^{\downarrow\text{Teaching}} = \{A, L\}$. Observe that a set of two objects $\{A, L\}$ and a set of two attributes $\{OS, DB\}$ correspond to each other and define each other uniquely in the context Teaching. So, in Teaching context $\{$A, L$\}$ is an example of a '*natural cluster*' of objects since this cluster is uniquely defined by a corresponding '*natural cluster*' of shared attributes $\{OS, DB\}$. In FCA terms, a pair ($\{$A, L$\}$, $\{$OS, BD$\}$) is called *formal concept* in a formal context Teaching. In contrast, a pair ($\{$A, L$\}$, $\{$OS, BD$\}$) is not a formal concept in the second formal context Research, since $\{$A, L$\}^{\uparrow\text{Research}}$ $= \{$OS$\} \neq \{$OS, DB$\}$ and $\{$OS, DB$\}^{\downarrow\text{Research}} = \emptyset \neq \{A, L\}$.

Now we are ready to introduce FCA realization problem at informal level. This time we are given some constrains for cross-tables and formal concepts expressed in set-theoretic and FCA terms. Then we are asked to generate cross-tables that match all given constrains (or refute a possibility to generate these cross-tables). The following staff-hiring problem could illustrate FCA realization problem.

Let us assume that another Computer Science Department would like to hire new staff, namely: 1 Full Professor (F), 1 Associate Professor (A), 1 Lecture (L), 2 Tutors (T_1 and T_2), and 1 Research Programmer (R). Teaching and research profile of new staff must cover Programming Languages (PL), Operating Systems (OS), Data Bases (DB), Formal Methods (FM). But due to department policy there are some requirements:

	Teaching					Research					Position					
	PL	OS	DB	FM	AI	PL	OS	DB	FM	AI	F	A	L	T_1	T_2	P
B	x	x	x			x	x				x	x				
D		x	x		x		x	x				x	x			
G	x		x		x	x								x	x	x
H		x		x	x		x	x		x				x	x	x
K	x			x	x				x	x	x					
L	x		x		x	x	x			x				x	x	x
M							x			x						x
N	x	x	x			x	x						x			

Table 2. Summary of sample applicants

1. Full Professor and Associate Professor together must cover all teaching subjects.
2. Lecturer must be able to teach some subject instead of Full Professor and some subject instead of Associate Professor.
3. Tutors together must cover all teaching subjects.
4. Every tutor must be able to teach a subject after Full Professor.
5. Full professor, Associate professor, and Lecturer must conduct research in some research areas within subject(s) they teach.
6. Research Programmer must support research of Full Professor.

Assume also that eight people have applied for these vacant positions: Beanka (B), Donald (D), Greta (G), Huan (H), Kevin (K), Leonora (L), Monica (M), and Norman (N). Summary of their applications is presented in the table 2. (It uses the same notation as table 1.)

A particular example of the staff-hiring problem is: whether it is possible to select five people among eight applicants to fill vacant positions and to meet six above requirements simultaneously?

Fortunately, this problem can be encoded easily as the following instance of the FCA-realization problem.

1. Split table 2 onto three formal contexts Teaching, Research, and Position. Present cross-tables of these formal contexts row by row as follows:
   – $B^{\uparrow \text{Teaching}} = \{\text{PL, OS, DB}\}$, $B^{\uparrow \text{Research}} = \{\text{PL, OS}\}$, $B^{\uparrow \text{Position}} = \{\text{F, A}\}$;
   – $D^{\uparrow \text{Teaching}} =$ .... , $D^{\uparrow \text{Research}} =$ .... , $D^{\uparrow \text{Position}} =$ .... ;
   – ....
2. Let U, V, X, Y_1, Y_2, and Z be variables for applicants to be appointed as Full Professor, Associate Professor, Lecture, Tutor_1 and Tutor_2, and Research Programmer:
   – $\{U, V, X, Y_1, Y_2, Z\} \subset \{B, D, G, H, K, L, M, N\}$,
   – $F \in U^{\uparrow \text{Position}}$, $A \in V^{\uparrow \text{Position}}$, $L \in X^{\uparrow \text{Position}}$,
   $T_1 \in Y_1^{\uparrow \text{Position}}$, $T_2 \in Y_2^{\uparrow \text{Position}}$, $P \in Z^{\uparrow \text{Position}}$.
3. Formulate position requirements in FCA terms using variables U, V, X_1, X_2, Y, and Z instead of position names F, A, L, T_1, T_2, and P:

  - $\{\mathrm{U}\}^{\uparrow\mathrm{Teaching}} \cup \{\mathrm{V}\}^{\uparrow\mathrm{Teaching}} = \{\mathrm{PL, OS, DB, FM, AI}\}$,
  - $\{\mathrm{X}\}^{\uparrow\mathrm{Teaching}} \cap \{\mathrm{U}\}^{\uparrow\mathrm{Teaching}} \neq \emptyset$, $\{\mathrm{X}\}^{\uparrow\mathrm{Teaching}} \cap \{\mathrm{V}\}^{\uparrow\mathrm{Teaching}} \neq \emptyset$,
  - ....

4. Find 'values' for variables U, V, X, Y_1, Y_2, and Z that meet all above constrains simultaneously, or refute existence of these values.

Fortunately, FCA realization problem is decidable due to a reduction to satisfiability problem for a description logic $\mathcal{ALBO}$ [4]. It implies that a particular instance of stuff-hiring problem as well as any other instance of this problem can be solved automatically by some tableau algorithm.

A reduction can be done by integrating FCA constructs into Description Logic (DL) [5]. For every particular description logic $\mathcal{L}$ let $\mathcal{L}$/FCA be an extension of $\mathcal{L}$ by FCA operations '$\uparrow$' and '$\downarrow$'. Paper [5] 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.

The paper is organized as follows. The next section 2 introduces basic notions and some facts related to Formal Concept Analysis. Integration of FCA into DL is sketched in the section 3. Formalization of FCA-realization problem and its reduction to $\mathcal{ALBO}$-satisfiability are presented in the last section 4.

2 Foundations of FCA

Basic Formal Concept Analysis (FCA) definitions below follow monograph [2], but we use a little bit different notation.

Definition 1. *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.*

For example, every terminological interpretation (D, I) defines a family of formal contexts $(D, D, I(R))$ indexed by role symbols and/or role terms R.

Usually formal contexts are represented as cross-tables. A cross-table for a formal context (O, A, B) is a matrix, where rows correspond to object, columns – to attributes; for any object s and attribute t, if $(s, t) \in B$ then a cross 'x' is located in a position (s, t) of the matrix. See table 1 for an example.

Two major algebraic operations for formal contexts are upper and lower derivations. They are used to define a notion of a formal concept.

Definition 2. *Let (O, A, B) be a formal context.*

- *For every set of objects $X \subseteq O$ its upper derivation $X^{\uparrow}$ is a 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 a 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 have simultaneously all attributes in Y.*

– *A formal concept is any 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 intent respectively.*

The above definition is given for a fixed single context when one can use a superscript notation '$\uparrow$' and '$\downarrow$' to denote upper and lower derivative operations for an implicit binary relation. But in the case when we have a family of formal contexts $K_j = (O_j, A_j, B_j)$, $j \in J$ it makes sense to attach an explicit reference K_j, or B_j, or index j to the superscripts, for example: '$\uparrow K_j$', '$\downarrow B_j$' or '$\uparrow j$'.

Definition 3. *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* $FC(K)$:
$$(Ex', In') \preceq_K (Ex'', In'') \text{ iff } Ex' \subseteq Ex'' \text{ and/or}^{2}\ In'' \subseteq In';$$
– *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 [2].

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 4. *For every formal context* K *let concept lattice* $CL(K)$ *be the following complete lattice* $(FC(K), \preceq_K, \top_K, \bot_K, \sup_K, \inf_K)$.

3 Integrating FCA operations into DL

There are many description logics, but we define in brief only some of them. Please refer a comprehensive handbook [1] for full details.

Attribute Language with Complements ($\mathcal{ALC}$) [3] 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. A formal syntax definition follows.

Definition 5. $\mathcal{ALC}$ *is a description logic which only role terms are role symbols* RS *and concept terms are defined by the following context-free grammar:*

$$C_{\mathcal{ALC}} ::= CS | \top | \bot | (\neg C_{\mathcal{ALC}}) | (C_{\mathcal{ALC}} \sqcup C_{\mathcal{ALC}}) | (C_{\mathcal{ALC}} \sqcap C_{\mathcal{ALC}}) | (\forall RS.\ C_{\mathcal{ALC}}) | (\exists RS.\ C_{\mathcal{ALC}})$$

[2] These two conditions are equivalent each other.

where metavariables CS and RS represent any concept and role symbols, respectively. Semantics of $\mathcal{ALC}$ is defined in the standard way in terminological interpretations [1].

Many description logics can be defined as extensions of $\mathcal{ALC}$ by concept and role constructs. Let us represent a collection of concept and role constructs by a sequence of concept constructs followed by a sequence of role constructs separated by a delimiter &[3]. So, a general pattern for a collection is $C\&R$, where C stays for a sequence of concept constructs, and R stays for a sequence of role constructs. For any collection of this kind let $\mathcal{ALC}(C\&R)$ be a 'closure' of $\mathcal{ALC}$ that admits all concept and role constructs in $C\&R$.

Definition 6. *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 '$\exists$'.*

Let us remark that '*variant*' is not a conventional term. We introduce it for technical convenience, since conventional DL glossary does not provide an equivalent term. Let us also remark that from the viewpoint of the above definition, $\mathcal{ALC}$ is the smallest variant of DL.

Definition 7. *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 example, $\mathcal{ALC}(\neg, -)$ is an extension of $\mathcal{ALC}$ where any role symbol can be negated and/or inverted. Another important example is Attribute Language with Boolean algebras on concepts and roles with Object symbols $\mathcal{ALBO}$ [4]. This description logic can be defined as $\mathcal{ALC}(OS\ \&\ ?, \neg, -)$, where '$OS$' represents a set of singleton concept term constructs '$\{a\}$' for every object symbol $a \in OS$ (that are called nominals[4]), and '?' represents two role term constructs – domain '$\upharpoonright$' and range '$\downharpoonright$' restrictions.

Fact 2 *Satisfiability and consistency problems are decidable for $\mathcal{ALBO}$ and there exist a tableau procedures that solve them. But (unfortunately) both problems are hard to solve: they are complete in the class of non-deterministic exponential time computations* **NExp-Time***.*

Please refer [4] for details.

Definition 8. *Let $\mathcal{L}$ be a variant of DL. Then let $\mathcal{L}/FCA$ be a closure of $\mathcal{L}$ with respect to two new formula constructors for the upper and lower derivatives. Syntax of these two constructs follows: for every role term R and every concept term X let $(X^{\uparrow R})$ and $(X^{\downarrow R})$ be concept terms too. They are read as 'upper derivative of X with respect to R' and, respectively, as 'lower derivative of X with respect to R'. For every terminological interpretation (D, I) let*

[3] Recall that DL does not use conjunction symbol '&', but intersection '$\sqcap$'. We drop out the delimiter '&' when concept/role constructs are implicit.

[4] Due to modal logic tradition. They are called constants n program logic tradition.

- $I(X^{\uparrow R}) = \{t :$ *for every* $s \in D$, *if* $s \in I(X)$ *then* $(s,t) \in I(R)\}$
  (– i.e. the upper derivation of $I(X)$ *in a formal context* $(D, D, I(R))$*);*
- $I(X^{\downarrow R}) = \{s :$ *for every* $t \in D$, *if* $t \in I(X)$ *then* $(s,t) \in I(R)\}$
  (– i.e. the lower derivation of $I(X)$ *in a formal context* $(D, D, I(R))$*).*

The definition states, that for every DL variant $\mathcal{L}$, $\mathcal{L}$/FCA is simply $\mathcal{L}(\uparrow,\downarrow)$. In particular, $\mathcal{ALC}$/FCA is an extension of $\mathcal{ALC}$ where both derivative constructors are allowed.

The following fact has been proved in Proposition 1 in [5].

Fact 3

1. *Let $\mathcal{L}$ be a variant of DL. For every role and concept terms within $\mathcal{L}$ the following concepts of $\mathcal{L}$/FCA and $\mathcal{L}(\neg,-)$ are equivalent:*
   (a) $\neg(X^{\uparrow R})$ *and* $\exists\neg R^{-}.\ X$*,*
   (b) $X^{\uparrow R}$ *and* $\forall\neg R^{-}.\ \neg X$*,*
   (c) $\neg(X^{\downarrow R})$ *and* $\exists\neg R.\ X$*,*
   (d) $X^{\downarrow R}$ *and*[5] $\forall\neg R.\ \neg X$*.*
2. *$\mathcal{L}$/FCA can be expressed in $\mathcal{L}(\neg,-)$ with linear complexity, i.e. every concept X in $\mathcal{L}$/FCA is equivalent to some concept Y in $\mathcal{L}(\neg,-)$ that can be constructed in linear time.*

In particular, $\mathcal{ALC}$/FCA is expressible in $\mathcal{ALC}(\neg,-)$, and $\mathcal{ALBO}$/FCA is expressively equivalent to $\mathcal{ALBO}$ itself. The decidability of $\mathcal{ALBO}$ (see Fact 2) implies the next corollary.

Corollary 1. *For any fragment $\mathcal{L}$ of $\mathcal{ALBO}$ the satisfiability and consistency problems for $\mathcal{L}$/FCA are decidable.*

4 FCA Realization Problem

Let us define a new algorithmic problem for Formal Concept Analysis which we call *FCA realization problem.* It can be defined informally as follows. Let us assume that Σ 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 with aid of set-theoretic operations, upper and lower derivative. The question is: can any set of formal contexts realize this 'collection' Σ?

Formal definitions follows. In these definitions we assume that

- Θ is a fixed set of object constants,
- Λ is a fixed set of attribute constants,
- Γ is a fixed alphabet of formal context names,
- Φ is a fixed alphabet of names for sets of objects,
- Ψ is a fixed alphabet of names for sets of attributes,

[5] Let us remark that 'operator' $\forall\neg R.\ \neg X$ has been already know in modal, program, and description logic community as 'window operator' [4].

– and Ω is a fixed alphabet of names for formal contexts.

Definition 9. *FCA-expression can be of one of two types: either of OS-type (object-set type) or of AS-type (attribute-set type). FCA-expressions and their types are defined by mutual recursion as follows.*

Object-set type expressions:

– *Every finite subset of Θ is an expression of OS-type.*
– *Every name in Φ is an expression of OS-type.*
– *For every name $\omega \in \Omega$, object set O_ω is an expression of OS-type.*
– *For every expression ε of AS-type and every context name $\omega \in \Omega$, lower derivative $\varepsilon^{\downarrow\omega}$ is an expression of OS-type.*
– *Every set-theoretic expression constructed from any expressions of OS-type by means of union '$\cup$', intersection '$\cap$' and set-difference '$\setminus$', is also an expression of OS-type.*

Attribute-set type expressions:

– *Every finite subset of attributes Λ is an expression of AS-type.*
– *Every name in Ψ is an expression of AS-type.*
– *For every name $\omega \in \Omega$, attribute set A_ω is an expression of AS-type.*
– *For every expression ε of OS-type and every context name $\omega \in \Omega$, upper derivative $\varepsilon^{\uparrow\omega}$ is an expression of AS-type.*
– *Every set-theoretic expression constructed from any expressions of AS-type by means of union '$\cup$', intersection '$\cap$' and set-difference '$\setminus$', is also an expression of AS-type.*

Definition 10. *A FCA-sentence has on of the following forms:*

– *$o \in \varepsilon$, where $o \in \Theta$ is any object, ε is any expression of OS-type;*
– *$a \in \varepsilon$, where $a \in \Lambda$ is any attribute, ε is any expression of AS-type;*
– *$\varepsilon' \subseteq \varepsilon''$, where ε' and ε'' are any FCA-expressions of one type;*
– *$(\varepsilon', \varepsilon'')FC(\omega)$ where ε' is any expression of OS-type, ε'' is any expression of AS-type, and $\omega \in \Omega$ is any context name.*

A FCA-system (of sentences) is any Boolean combination of FCA-sentences.

Definition 11. *A valuation is a mapping υ that assigns*

– *a set $\upsilon(\iota)$ to every name $\iota \in \Phi$,*
– *a set $\upsilon(\kappa)$ to every name $\kappa \in \Psi$*
– *a formal context $\upsilon(\omega) = (O_{\upsilon(\omega)}, A_{\upsilon(\omega)}, B_{\upsilon(\omega)})$ to every name $\omega \in \Omega$,*

in such a way that the following conditions hold:

– *for set of objects $\Theta \subseteq \cup_{\omega\in\Omega} O_{\upsilon(\omega)}$;*
– *for set of attributes $\Lambda \subseteq \cup_{\omega\in\Omega} A_{\upsilon(\omega)}$;*
– *for every name $\iota \in \Phi$, $\upsilon(\iota) \subseteq \cup_{\omega\in\Omega} O_{\upsilon(\omega)}$;*
– *for every name $\kappa \in \Psi$, $\upsilon(\kappa) \subseteq \cup_{\omega\in\Omega} A_{\upsilon(\omega)}$.*

Let us extend valuations on FCA-expressions in the following natural way.

Definition 12.

Object-set type expressions:

– $\upsilon(S) = S$ *for every finite subset* S *of* Θ.

– $\upsilon(\varepsilon^{\downarrow\omega}) = (\upsilon(\varepsilon))^{\downarrow\upsilon(\omega)}$ *for every AS-type expression* ε *and name* $\omega \in \Omega$.

Attribute-set type expressions:

– $\upsilon(S) = S$ *for every finite subset* S *of* Λ.

– $\upsilon(\varepsilon^{\uparrow\omega}) = (\upsilon(\varepsilon))^{\uparrow\upsilon(\omega)}$ *for every OS-type expression* ε *and name* $\omega \in \Omega$.

Both FCA types:

$\upsilon(\varepsilon' \cap \varepsilon'') = \upsilon(\varepsilon') \cap \upsilon(\varepsilon'')$,
$\upsilon(\varepsilon' \cup \varepsilon'') = \upsilon(\varepsilon') \cup \upsilon(\varepsilon'')$,
$\upsilon(\varepsilon' \setminus \varepsilon'') = \upsilon(\varepsilon') \setminus \upsilon(\varepsilon'')$.

The following proposition is easy to prove by induction on structure of FCA-Expressions.

Proposition 1. *For every valuation* υ*, for every OS-type and AS-type FCA-expressions* ϕ *and* ψ *respectively* $\upsilon(\phi) \subseteq \cup_{\omega\in\Omega} O_{\upsilon(\omega)}$*, and* $\upsilon(\psi) \subseteq \cup_{\omega\in\Omega} A_{\upsilon(\omega)}$.

This proposition leads to opportunity to extend valuations onto FCA-sentences.

Definition 13. *Every evaluation* υ *assigns Boolean value 'true' or 'false' to FCA-sentences as follows:*

1. $\upsilon(o \in \varepsilon) = true$ *if* $o \in \upsilon(\varepsilon)$*; otherwise* $\upsilon(o \in \varepsilon) = false$*;*
2. $\upsilon(a \in \varepsilon) = true$ *if* $a \in \upsilon(\varepsilon)$*; otherwise* $\upsilon(a \in \varepsilon) = false$*;*
3. $\upsilon(\varepsilon' \subseteq \varepsilon'') = true$ *if* $\upsilon(\varepsilon') \subseteq \upsilon(\varepsilon'')$*; otherwise* $\upsilon(\varepsilon' \subseteq \varepsilon'') = false$*;*
4. $\upsilon((\varepsilon', \varepsilon'')FC(\omega)) = true$ *if* $(\upsilon(\varepsilon') , \upsilon(\varepsilon'')$ *is a formal concept in a formal context* $\upsilon(\omega)$*; otherwise* $\upsilon((\varepsilon', \varepsilon'')FC(\omega)) = false$.

We are ready to define satisfiability for FCA-systems and FCA realization problem.

Definition 14. *For every evaluation* υ *and every FCA-system (of sentences)* Σ *let* $\upsilon(\Sigma)$ *be a Boolean value 'true' or 'false' computed according to the standard rules from values of FCA-sentences. An FCA-system* Σ *is said to be satisfiable if there exists an evaluation* υ *such that* $\upsilon(\Sigma) = true$*. FCA realization problem is an algorithmic problem to check for input FCA-system whether it is satisfiable or not.*

Proposition 2. *FCA realization problem is decidable.*

Proof. Let Σ be an input FCA-system. Since Σ is a Boolean combination of FCA-sentences, it can be transformed to equivalent system but in a disjunctive normal form $\bigvee_{i\in\ldots}(\bigwedge_{j\in\ldots}\pi_{(i,j)})$ where all $\pi_{(i,j)}$ are FCA-sentences or their negations. Hence Σ is satisfiable iff any conjunction $(\bigwedge_{j\in\ldots}\pi_{(i,j)})$ is satisfiable. So we may assume that Σ is a conjunction of FCA-sentences or their negations.

But FCA realization for a system of this kind is easy to reduce to consistency of $\mathcal{ALBO}$ knowledge base, that is a known decidable problem (Fact 2). A variant of reduction is presented below.

Let alphabet of object symbols consists of all elements in $\Theta \cup \Lambda$, Let alphabet of concept symbols consists of all O_ω and A_ω for all $\omega \in \Omega$. Let alphabet of role symbols consists of all B_ω for all $\omega \in \Omega$. Observe that in this syntax every FCA-expression can be considered as a concept term[6]. The every FCA-sentence or its negation can be converted to equivalent terminological or assertional sentences as follows:

- $o \in \varepsilon \ \mapsto \ o : (\sqcup_{\omega\in\Omega} O_\omega) \ \sqcap \ \varepsilon$;
  $\neg(o \in \varepsilon) \ \mapsto \ o : (\sqcup_{\omega\in\Omega} O_\omega) \ \sqcap \ (\neg\varepsilon)$;
- $a \in \varepsilon \ \mapsto \ a : (\sqcup_{\omega\in\Omega} A_\omega) \ \sqcap \ \varepsilon$;
  $\neg(a \in \varepsilon) \ \mapsto \ a : (\sqcup_{\omega\in\Omega} A_\omega) \ \sqcap \ (\neg\varepsilon)$;
- $\varepsilon' \subseteq \varepsilon'' \ \mapsto \ \varepsilon' \ \dot{\sqsubseteq} \ \varepsilon''$; $\neg(\varepsilon' \subseteq \varepsilon'') \ \mapsto \ x : (\varepsilon' \sqcap \neg(\varepsilon''))$, where '$x$' is fresh object symbol;
- $(\varepsilon', \varepsilon'')FC(\omega) \ \mapsto \ (\varepsilon' \ \dot{\sqsubseteq} \ O_\omega, \ \varepsilon'' \ \dot{\sqsubseteq} \ A_\omega, \ (\varepsilon')^{\uparrow B_\omega} \doteq \varepsilon''), \ (\varepsilon'')^{\downarrow B_\omega} \doteq \varepsilon')$.

In this way we can convert any conjunction of FCA-sentences and their negations to some $\mathcal{ALBO}$/FCA knowledge base which equivalent with respect to satisfiability. But $\mathcal{ALBO}$/FCA is $\mathcal{ALBO}(\uparrow, \downarrow)$ which in turn is expressible in $\mathcal{ALBO}(\neg, -) = \mathcal{ALBO}$. But $\mathcal{ALBO}$-consistency problem is decidable (see Fact 2). It finishes the proof. ■

Unfortunately, our decision procedure (that is sketched in the proof of proposition 2) is extremely inefficient and impractical: we reduce the problem to exponential number of instances of $\mathcal{ALBO}$-satisfiability problem, which is known to be **NExp-Time**-complete [4].

References

1. 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.
2. Ganter B., Wille R. *Formal Concept Analysis.* Mathematical Foundations. Springer Verlag, 1996.
3. Schmidt-Schauß M., and Smolka G. *Attributive concept descriptions with complements.* J. Artificial Intelligence, Vol. 48, 1991, pp. 1-26.
4. Schmidt R.A., Tishkovsky D. *Using Tableau to Decide Expressive Description Logics with Role Negation.* The Semantic Web. Proceedings of 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference. Springer Lecture Notes in Computer Science, v.4825, 2007, p.438-451.
5. Shilov N.V. Garanina N.O., Anureev I.S. *Combining Two Formalism for Reasoning about Concepts.* Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen Italy, 2007. CEUR Workshop Proceedings v.250. p.459-466.

[6] It is sufficient to replace '$\cap$', '$\cup$' and '$\setminus$' by '$\sqcap$', '$\sqcup$' and '$\sqcap\neg$'.