# A Semi-automatic Method to Ontology Design by Using FCA

Hele-Mai Haav

Institute of Cybernetics at Tallinn University of Technology
Akadeemia tee 21, 12618 Tallinn, Estonia
helemai@cs.ioc.ee

**Abstract.** Ontology design is a complex and time-consuming process. It is extremely difficult for human experts to discover ontology from given data or texts. This paper presents a semi-automatic method for ontology extraction and design. The method is based on Formal Concept Analysis and a Horn clause model of a concept lattice. Inputs to the technique are domain-specific texts or data. After transformations, resulting domain-specific ontology is represented as a set of rules and facts according to Horn clause model of concept lattice based ontology representation. Ontology designer is given this initial ontology expression for further extension by adding concepts and relationships (part-of, related to, etc) by using a rule language based on Horn clauses. Validation of ontology is done by logical inference.

## 1 Introduction

Manual construction and description of domain-specific ontology is a complex and time-consuming process. The recent study on ontology design methodologies shows that it is very hard for a designer to develop accurate and consistent ontology [10].

Therefore, a number of approaches propose to improve ontology construction using automatic discovery of taxonomic and non-taxonomic relationships from domain data or domain-specific texts [1, 4, 8, 9, 12]. These approaches can be classified to two groups: most of the approaches provide methods for automatic discovery of taxonomic relationships and only few of them deal with learning ontological relationships between concepts. Unfortunately, in the approaches available, there is a lack of combination of the two methods, because methods for learning ontological relationships rely to a given initial taxonomy of concepts and use it in learning process [8, 11].

The goal of this paper is to present a new approach to semi-automatic ontology extraction and design by using Formal Concept Analysis (FCA) [2] combined with a rule-based language. Our approach provides tools for automatic or semi-automatic extraction of taxonomy of concepts from domain-specific texts, automatic transformation this initial ontology to Horn clause language and a rule language for expression of non-taxonomic relationships. Validation of ontology is done by logical inference.

© V. Snášel, R. Bělohlávek (Eds.): CLA 2004, pp. 13–24, ISBN 80-248-0597-9.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2004.

Our approach is applicable in many application domains, where domain specific ontologies can be extracted from web catalogs, product catalogs, domain specific dictionaries and texts etc.

The rest of the paper is structured as follows. Section 2 briefly overviews related works. Two basic notions used in our approach: concept lattice based ontology expression and its first-order logic model are introduced in sections 3 and 4. General framework of proposed approach is presented in section 5. Section 6 concludes the work.

## 2 Related Works

There are several attempts on ontology learning and ontology extraction [1, 7, 8, 9, 11, 12]. For example, in KRAFT [9], local ontology is extracted from shared ontology. In the system Infosleuth [12] ontologies are constructed semi-automatically from textual databases. They use ontology-learning approach, where human experts provide a set of initial words denoting concepts from high-level ontology. In [8] discovering non-taxonomic relationships from texts using shallow text processing methods is presented. Their technique is integrated into ontology learning tool TextToOnto. The method presented in [11] uses manually built initial ontology to build conceptual hierarchy semi-automatically from texts. Resulting ontology is presented in Description Logics.

Similarity of our approach and those discussed above is in using texts as descriptions of conceptualisation of domain and learning formal ontology from the given texts.

To contrast our approach to the research mentioned above, we like to express that we combine automatic extraction of taxonomic relationships from domain-specific texts with semi-automatic expression of full ontology (including also non-taxonomic relationships) in first-order language for further reasoning and search.

Before starting to present our framework for ontology extraction, we first introduce two basic notions of our method: concept-lattice based ontology representation and its first order language model.

## 3 Concept Lattice Based Ontology Representation

In this section, we define concept lattice based ontology expression as a reduced concept lattice of a given context K(O,C,R), where O is object set, C is set of attributes of objects and R is a binary relationship between objects and attributes. Our definition is based on FCA [2], which is lattice theory applied to the analysis of hierarchies of concepts. We assume that a reader of this paper is briefly familiar with this theory. Most important FCA notions used in the paper will be discussed below by illustrative examples. We defined concept lattice based ontology representation first in [4]. In this paper we refine this definition to meet requirements of the method presented.

### 3.1 A Brief Introduction to FCA and to Sample Domain

A reader is referred to [2] for detailed knowledge about FCA. In the following we give a very basic introduction to the main principles of FCA by using examples.

For example, a context K(O,C,R) of real estate domain can be as shown as in Table 1. The set of objects O is a collection of real estate domain specific texts (ads) about real estate items denoted by references like A1, A2, etc. in the table 1. The set of attributes C is the set of noun-phrases chunked from these texts by using NLP. If a text describing a real estate item contains certain noun-phrase, then the relationship R holds and we denote it by number 1 in the table below.

**Table 1.** Real estate domain context

| Objects | Attributes (noun-phrases) | | | | | |
|---|---|---|---|---|---|---|
| | Real estate | Family house | Country house | Summer house | Blockhouse | Skyscraper |
| A1 | 1 | 1 | | | | |
| A2 | 1 | 1 | 1 | 1 | | |
| A3 | 1 | | 1 | | | |
| A4 | 1 | | | | 1 | 1 |
| A5 | 1 | | | | 1 | |
| A6 | 1 | | | 1 | | |

A formal concept of the context K(O, C, R) is defined as pair (A, B), where $A \subseteq O$, $B \subseteq C$, $A' = B$ and $B' = A$, where $A'$ is the set of attributes common to all the objects in A and $B'$ is the set of objects having the attributes in B. The extent of the concept (A, B) is A and its intent is B.

For concepts (A1, B1) and (A2, B2) in the set S of all concepts of the context K(O,C,R) we have

$$(A1, B1) \leq (A2, B2) \Leftrightarrow A1 \subseteq A2 \Leftrightarrow B1 \supseteq B2.$$

The relation $\leq$ is an order on S. It is shown in [2] that $(S(K), \leq)$ is a complete lattice and this lattice is known as the concept lattice of the context K(O, C, R).

For example, the concept lattice in Fig. 1 corresponds to the context presented in the table 1.

Each node in this lattice (denoted by black circle) is a formal concept. For example, one of the formal concepts of the context described in Table 1 is *{A2, A6} × {Real estate, Summerhouse},* where the set *{A2, A6}* is the extent of the concept and the set *{Real estate, Summerhouse}* is its intent.

Sub and super-concept relationships between the formal concepts are represented by edges in the Hasse diagram in Fig. 1. For example, the formal concept *{A4} × {Real estate, Blockhouse, Skyscraper}* is a sub-concept of the concept *{A4, A5} × {Real Estate, Blockhouse}.* There are no objects that are defined by all the attributes, so the extent of the bottom concept is empty.

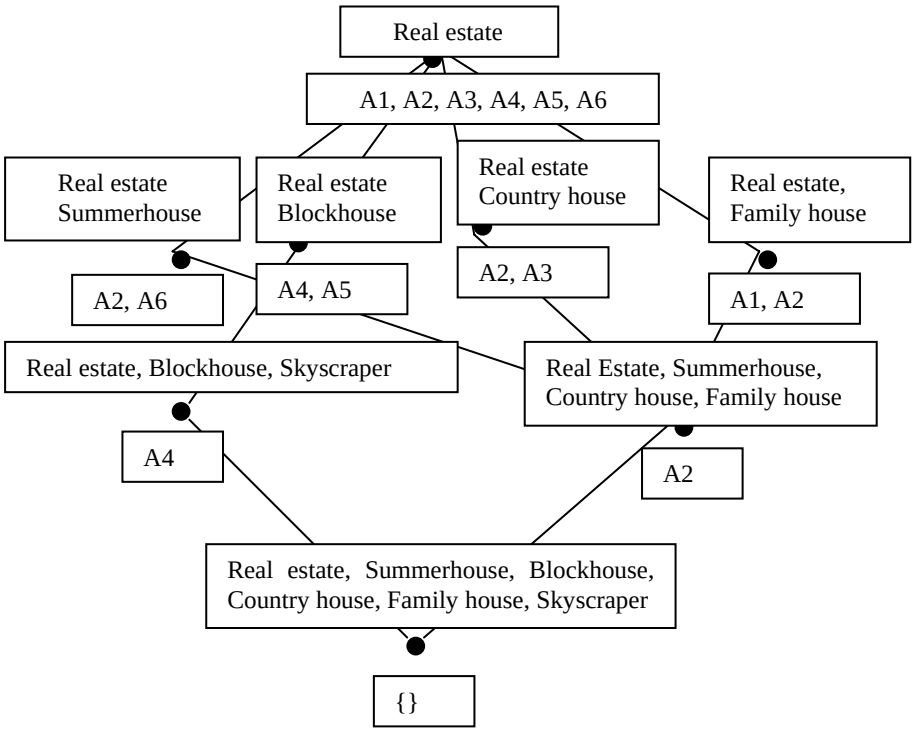


**Fig. 1.** The concept lattice of a real estate domain

### 3.2 Concept Lattice Based Ontology Expression

There is redundant information in concept lattice. The two kinds of redundancy can be eliminated from concept lattice without loosing any information as shown in [3]: redundant elements in formal concepts intents and redundant objects in formal concepts extents. Our reduction procedure has 2 steps: elimination of redundant elements from formal concepts intents and elimination of lattice of extents.

***Elimination of redundant elements***. For a pair (A, B), B (intent) will appear in every descendant. The inherited elements may be eliminated. Let B´ be the set of elements in B that do not appear in any descendant of B. Then we can consider the concept lattice, where the nodes contain the pairs (A, B´) instead of pairs (A, B). This lattice is a result of the first step of our reduction procedure.

***Elimination of lattice of extents***. After elimination of redundant elements from concept intents, we eliminate lattice of extents $L_O$ and get as a result a reduced lattice of intents $L_{CR}$ of formal concepts.

We call the resulting lattice $L_{CR}$ of reduction procedure as concept lattice based ontology expression. Fig. 2 shows the lattice $L_{CR}$ of our example.

***Naming concepts.*** After reducing concept lattice to its intentional part, we need to give formal concepts the names. Naming in our case can be done as follows:

1. A concept gets a unique name that is the name of the attribute(s) of formal concepts, which are left after reduction procedure. For each attribute c, there is a most general concept whose intent includes c, this is called *attribute concept* and the name of this concept is the name of corresponding attribute(s). Let us recall that attributes of formal concepts indicate domain specific concepts in our approach.
2. After the previous naming procedure, there might be nodes that do not get names. In principle, the names for these nodes need to be provided by domain expert or ontology designer. It is possible automatically generate formal names (e.g. c1, c2…) for those nodes and then ask advice from human expert.

The Fig. 2 depicts real estate domain ontology produced from concept lattice shown in Fig.1 using reduction and naming procedures.

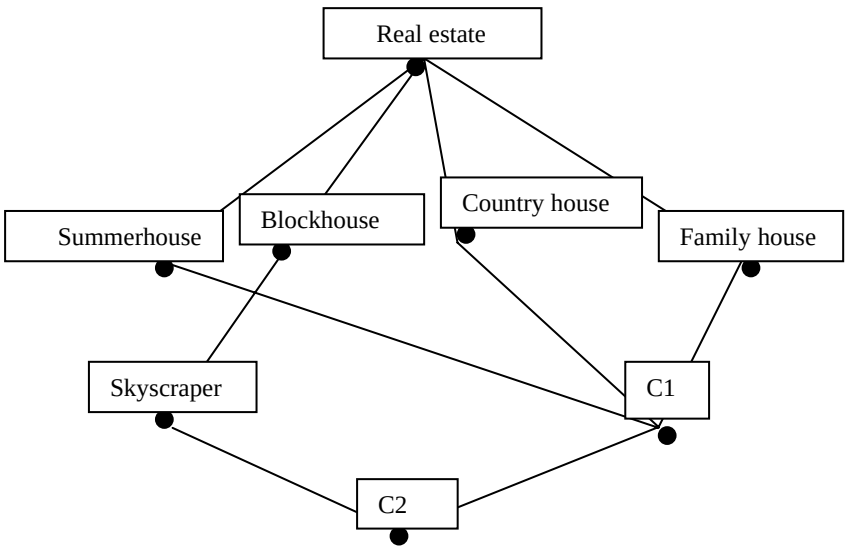


**Fig. 2.** Concept lattice based ontology representation

One may notice that 2 nodes in the lattice did not get names according to the naming procedure. The nodes denoted by generated concept names C1, C2 do not have labels in the lattice. Ontology designer may analyse the lattice above and give the appropriate names to the concepts. It is also interesting that those nodes really denote new unknown (discovered) concepts, because the domain-specific texts did not include any noun-phrases for denoting these concepts.

In principle, the lattice shown in figure 2 is just a complete lattice, which nodes are labeled by concept names. We call this lattice concept lattice based ontology representation because it is obtained from full concept lattice that is extracted by FCA from the given context.

# 4 A Logic Model of Concept Lattice Based Ontology Representation

In the previous section, we have defined domain ontology as a concept lattice reduced to its intensional part $L_{CR}$ (see for example Fig. 2). In this section, we provide first order logic model for $L_{CR}$. The model was first provided in [5]. At the moment we are not interested in extensions, which gave birth to full concept lattice for a formal context of a domain. Nevertheless, existence of full concept lattice gives always an opportunity to find extent of a given concept. In order to build first order logic model for concept lattice based ontology expression we need to define mappings from lattice structure to a first order language.

## 4.1 Language Constructs

We use standard syntax for first order logic and define a simple rule language based on Horn clauses as follows.

An alphabet of the rule-language is defined as follows:

1. Set of constants **N** that consists of the set of concept names **C**, names of properties A, and special names **any** (lattice top, empty top is always True) and **nil** (lattice bottom, empty bottom is always False).
2. Set of variable names **V**. Uppercase letters denote the variables in **V**.
3. Set of predicate symbols **P**

Terms are either constants or variables. An atom (atomic formula) is a formula of the form $p(t_1,...,t_n)$ , where p is a predicate symbol and $t_1,...,t_n$ are terms. A formula is called ground if it contains no variables.

Horn clause (rules) have at most one atom on its head and they are formulas of the following form:

$$A \leftarrow B_1, B_2,...,B_n ,$$

where $B_1, B_2,...,B_n$ is conjunction of atoms $B_i$, i=0,...,n and universal quantification of variables is assumed.

An interpretation for the rule-language is defined as a set of ground atoms constructed from predicate names in **P** and constants in **N.** As the language is general, then different inference engines can be used.

## 4.2 Transformation of Concept Lattice to Rule Language

The mappings from lattice to rule language are defined as follows.

***Mapping concepts***

Concepts are represented using their names in ontology as constants in rule language. For example, house, summerhouse etc. If we like to refer to extents, then concepts can also be represented by predicates like house(X). At the moment, we are not interested in extents of concepts.

***Mapping taxonomic relationships***

The predicate subconcept(Concept1,Concept2) is used to denote that *Concept1* is an immediate subconcept of *Concept2*. Subconcept predicates are automatically generated according to the given lattice $L_{CR}$.

Predicate isa is used to represent partial order relationship between concepts. For example, the predicate isa(summerhouse, real-estate) defines partial order relationship between the concepts summerhouse and real-estate stating that *summerhouse* isa *real-estate* (i.e. summerhouse is subconcept of real-estate ).

***Rules for lattice axioms***

As $L_{CR}$ is complete lattice, then the rules for lattice axioms are as follows:

*Reflexivity:*
isa(Concept, Concept)
*Transitivity:*
isa(Concept1, Concept2)←subconcept(Concept1, Concept2)
isa(Concept1, Concept2) ← isa(Concept1, Concept3), isa(Concept3, Concept2)
Predicate subconcept denotes that Concept1 is an immediate subconcept of Concept2.
*Antisymmetry:*
equal(Concept1,Concept2) ← isa(Concept1,Concept2), isa(Concept2,Concept1)

***Rules for lattice operations***

As $L_{CR}$ is a complete lattice, then for each set of concepts, there exists always a greatest lower bound (glb or greatest common subconcept) and a least upper bound (lub or a least common superconcept). Lattice meet is used to calculate glb and join is operation to calculate lub. We define these operations using the following set of rules.

*Lattice meet operation*
$c\_subconcept(C, C_1,\ldots,C_k) \leftarrow isa(C, C_1),\ldots, isa(C, C_k)$
$g\_c\_subconcept(C, C_1,\ldots,C_k) \leftarrow c\_subconcept(C, C_1,\ldots,C_k),$
$c\_subconcept(T, C_1,\ldots,C_k), isa(T, C)$

The predicate $c\_subconcept(C, C_1,\ldots,C_k)$ means that the concept C is a common subconcept of the set of concepts $\{C_1,\ldots,C_k\}$

The predicate $g\_c\_subconcept(C, C_1,\ldots,C_k)$ means that the concept C is the greatest common subconcept of the set of concepts $\{C_1,\ldots,C_k\}$.

Symmetrically, we define predicates and rules for join operation as follows:
*Lattice join operation*
$c\_superconcept(C, C_1,\ldots,C_k) \leftarrow isa(C_1, C),\ldots, isa(C_k, C)$
$l\_c\_superconcept(C, C_1,\ldots,C_k) \leftarrow c\_superconcept(C, C_1,\ldots,C_k),$
$c\_superconcept(T, C_1,\ldots,C_k), isa(C, T)$

Logical model of a given lattice $L_{CR}$ can automatically be generated on the basis of mappings presented above. This process is demonstrated in the following example.

### 4.3 Examples about Transformations

The following set of ground subconcept atoms is automatically generated on the basis of concept lattice based ontology expression shown in Fig. 2.

subconcept(familyhouse, real-estate)
subconcept(summerhouse, real-estate)
subconcept(countryhouse, real-estate)
subconcept(blockhouse, real-estate)
subconcept(c1, countryhouse)
subconcept(c1, summerhouse)
subconcept(c1, familyhouse)
subconcept(skyscraper, blockhouse)
subconcept(c2, c1)
subconcept(c2, skyscraper)

On the basis of this set of atoms, partial order relationships can be derived using predefined rules from the Horn clause model of concept lattice based ontology representation. Also, lattice operations can be performed and consistence of ontology expression can be checked.

## 5 The Ontology Design Method

The proposed semi-automatic ontology design method is based on the two concepts introduced in previous sections: concept lattice based ontology representation and its logic model.

### 5.1 General Schema of the Method

The method comprises the following steps:
1. Extracting formal context of a domain from domain-specific texts or data
2. Computing initial ontology as the concept lattice from the context using FCA and reduction procedures
3. Transforming initial ontology to a set of expressions in first order language
4. Extending initial ontology with additional rules and facts

General schema of this method is drawn in the Fig. 3 as follows:

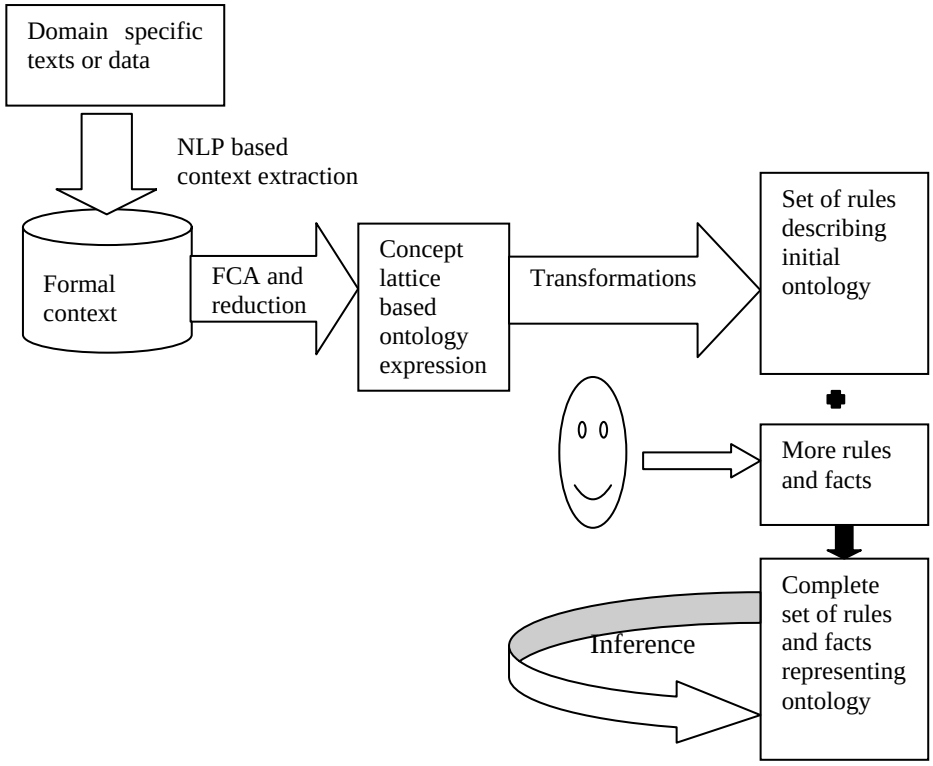


**Fig. 3**. General schema of the semi-automatic ontology design method

In the following we describe each step of the method in more detail.

### 5.2 Extraction of a Formal Context

According to the method, first task is to produce a formal context K(O,C,R) for extractable domain-specific ontology. For our approach, objects of formal context can be textual descriptions of domain entities written in some natural language. We assume that those descriptions use domain specific vocabulary and are rather short. For example, suitable descriptions can be ads of real estate items in the real estate web catalogues, descriptions of products in product catalogues, technical descriptions of components.

Attributes of an object for FCA are noun-phrases present in the domain-specific text describing a given domain-specific entity. Binary relationship R between descriptions (texts) of domain entities and noun phrases is discovered during the NLP process of text sources. A resulting set of noun phrases together with references to the domain-specific text sources are stored into the database table, which represents a context for the application domain in the form of binary relationship between descriptions of entities and noun phrases.

### 5.3 Computing Initial Ontology

In our approach, FCA is performed on the database, which stores formal context. After that the resulting concept lattice is reduced and certain naming procedure is performed in order to get concept lattice based ontology expression.

As formal concept lattice (even its reduced form) can be large, then a user should be given tools for browsing the lattice.

There exist several algorithms for FCA and construction of line diagrams of concept lattices. Excellent comparison of performance of those algorithms can be found in [6].

### 5.4 Transforming Initial Ontology to Horn Logic

Next step is to provide means for transforming concept lattice based ontology expression into Horn logic. This process enables to produce logical expression of ontology lattice and specify intended semantics of the descriptions in first order logic.

For that purpose, we introduced Horn logic based formulation of concept lattice based ontology representation in the section 4 of this paper. Initial ontology is automatically transformed to a set of facts according to this formulation. The latter includes also rules for partial order relationships, lattice axioms and lattice operations in order to reason about ontology. From the logical descriptions inference can automatically be done using one of automatic theorem provers. As our approach uses first order language, then it is possible to attach different ontology inference engines for practical applications by translating ontology expression to any inference engine rule language. This was one of the reasons behind choosing Horn logic based rule language.

### 5.5 Extending Initial Ontology with Additional Rules and Facts

As we have seen, taxonomic relationships between concepts can be automatically generated from a given lattice based ontology expression using logic-based formulation of concept lattice. In order to define non-taxonomic relationships the corresponding groups of predicates and rules are defined.

***Properties of concepts***

For defining properties of concepts, the following predicate can be used:
hasproperty(Conceptname, Propertyname).

***Inheritance of properties***

Inheritance of properties can be represented by the following rule:
hasproperty(C1, X)←isa(C1,C2), hasproperty(C2, X)

***Ontological relationships***

Ontological relationships like part-of, related-to etc can be easily represented via predicates. The following predicates demonstrate opportunities adding other ontological relationships:
partof(C1,C2)

related(C1,C2)
synonyms(C1,C2), etc.

Additional specific inference rules can easily be added to the set of predefined lattice based inference rules by ontology designer.

Non-taxonomic relationships give additional possibilities for ontology representation and reasoning about ontology.

### 5.6 Reasoning about Ontology

Reasoning is important to ensure the quality of design of ontology. It can be used to find contradictory concepts, to derive implied relationships, etc.

Inference rules for lattice axioms and operations can be used to decide taxonomic relationships between concepts as well as perform lattice operations.

For example, to find the least common superconcept of the set of concepts *{summerhouse; countryhouse},* we define the following query:

l_c_superconcept(X, summerhouse, countryhouse).

The answer is the concept *real-estate*. If we are interested in finding the greatest common subconcept of the concepts *familyhouse* and *countryhouse*, then the corresponding query is as follows: g_c_subconcept(X, familyhouse, countryhouse). The answer is the concept *c1*.

We may be interested in all the superconcepts of the concept *skyscraper*, for example. The query isa(skyscraper, X) gives the list of ground atoms as an answer. In our example, this is the list of the following atoms isa(skyscraper, real-estate) and isa(skyscraper, blockhouse).

Inference about non-taxonomic relationships is made possible due to additional rules. For example, a designer may add the fact hasproperty(blockhouse,no_of_floors). Using inference, we may ask query hasproperty(X, no_of_floors) and receive an answer that also the fact hasproperty(skyscraper, no_of_floors) holds.

Logical inference about ontology expression provides means for designing consistent and accurate ontologies.

## 6 Conclusion

We have provided a method to help ontology designer automatically to extract initial ontology from given set of domain specific texts, to map it automatically to rule language and to use rule language for adding non-taxonomic relationships to the ontology representation. Giving means for reasoning about ontology expression ensures consistency and accuracy of designed ontology.

Our main contribution resides in extracting the ontology from the NL texts by using FCA and transforming it to Horn logic.

Our future work is concerned about evaluation of the proposed method by developing some prototypical tool. By now, we only made experiments on different components of the approach.

## Acknowledgements

This research was partially funded by Enterprise Estonia funding as a part of RQL project.

## References

1. Decker, S., Erdmann, M., Fensel, D., and Studer, R.: Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information, Meersman R. et al. (eds.): Semantic Issues in Multimedia Systems. Proceedings of DS-8. Kluwer Academic Publisher, Boston, (1999), 351-369
2. Ganter, B. and Wille, R.: Formal Concept Analysis, Mathematical Foundations, Springer, (1999)
3. Godin, R., Missaoui, R. and Alaoui, H.: Learning Algorithms Using a Galois Lattice Structure, Proc. of the Third Int. Conference on Tools for Artificial Intelligence, IEEE Computer Society Press, CA, (1991), 22-29
4. Haav, H.-M.: Learning Ontologies for Domain-specific Information Retrieval, W. Abramowicz (Ed), Knowledge-Based Information Retrieval and Filtering from the Web, Kluwer Academic Publishers, (2003), 285-300
5. Haav, H.-M.: Combining FCA and a Logic Language for Ontology Representations, J. Barzdins (Ed), Databases and Information Systems, Sixth International Baltic Conference on Databases and Information Systems, June 6-9, Riga, 2004, Scientific papres of University of Latvia, Vol 672, (2004), 436-451
6. Kuznetsov, S.O. and Obiedkov, S.A.: Comparing performance of Algorithms for Generating Concept Lattices, Journal of Experimental & Theoretical Artificial Intelligence vol 14(1), (2002), Nos. 2-3, 189-216
7. Maedche, A.: Ontology Learning for the Semantic Web, Kluwer Academic Publishers, (2002)
8. Maedche, A. and Staab, S.: Discovering Conceptual Relations from Text. ECAI 2000. Proceedingsof the 14th European Conference on Artificial Intelligence, IOS Press, Amsterdam, (2000)
9. Preece, A.D., Hui, K.-J., Gray, W.A., Marti, P. et al: The Kraft arhitecture for knowledge fusion and transformation, In Proc. of 19th SGES Int. Conference, Springer-Verlag (1999)
10. Tempich, C. and Volz, R.: Towards a benchmark for Semantic Web reasoners-an analysis of the DAML ontology library, Sure Y (ed) Proceedings of Workshop of Evaluation of Ontology-based Tools (EON 2003) at 2nd Int. Semantic Web Conference (ISWC 2003), USA, (2003)
11. Todirascu, A. and Beuvron, F.: Using Description Logics for Ontology Extraction, In: Proceedings of Conference TALN 2000, Lausanne, October 16-18, (2000)
12. Woelk, D. and Tomlinson, C.: The Infosleuht project: Intelligent Search Management via Semantic Agents. In Second World Wide Web Conference ’94:Mosaic and the Web, (1994)