Ontology Design with Formal Concept Analysis

Marek Obitko[1], Václav Snášel[2], and Jan Smid[3]

[1]Department of Cybernetics, Czech Technical University in Prague, Czech Republic
[2]Department of Computer Science, VŠB–TU Ostrava, Czech Republic
[3]Computer Science Department, Morgan State University, Baltimore MD, USA
[1]obitko@labe.felk.cvut.cz, [2]vaclav.snasel@vsb.cz, [3]jsmid@jewel.morgan.edu

Abstract. Ontologies, often defined as an explicit specification of conceptualization, are necessary for knowledge representation and knowledge exchange. Usually this means that ontology describes concepts and relations that exist in a domain. To enable knowledge exchange, it is necessary to describe these concepts and relations in a better way than just ordering them in taxonomy. However, ontology design usually starts and stops with designing taxonomies. We present a method that is based on formal concept analysis, which is a theory of data analysis which identifies conceptual structures among data sets. This method allows for discovering necessity for new concepts and relations in an ontology, which leads to an ontology that has these entities described in a way suitable for knowledge exchange.

1 Introduction

Ontologies, often defined as an explicit specification of conceptualization [4], are necessary for knowledge representation and knowledge exchange [6]. Usually this means that ontology describes concepts and relations that exist in a domain [5]. To enable knowledge exchange, it is necessary to describe these concepts and relations in a better way than just ordering them in taxonomy. For example the concept should be described not only by its position in the taxonomical (is-a) hierarchy, but also e.g. by relations that can be applied to the concept. Similarly, the relation can be described by concepts that can be related together by this relation. However, ontology design usually starts and stops with designing taxonomies. Taxonomies are important, since they form "backbone" of an ontology, but are not enough for knowledge sharing.

We present a method for designing ontologies that is based on formal concept analysis [3]. Formal concept analysis (FCA) is a theory of data analysis which identifies conceptual structures among data sets. This ontology design method allows for discovering necessity for new concepts and relations in an ontology, which leads to an ontology that has these entities described in a way suitable for knowledge exchange or for information retrieval [7].

The rest of this paper is organized as follows: In the next section, we describe ontologies in general, and then we describe the formal concept analysis. In the following section, we describe our proposed method for ontology design, which is

© V. Snášel, R. Bělohlávek (Eds.): CLA 2004, pp. 111–119, ISBN 80-248-0597-9.
VŠB – Technical University of Ostrava, Dept. of Computer Science, 2004.

illustrated in detail in the next section. After that, we show how to map the result to ontology languages used today. We wrap up the paper with conclusion.

2 Ontologies

As we already mentioned, ontologies are usually defined as an explicit specification of conceptualization [4]. Explicit specification of conceptualization means that ontology is a description of the concepts and relationships that exist in a domain.

Other similar definitions are available (see [6] for comparison and discussion). Although they are not exactly identical, they in principle say that any ontology consists of the conceptualization of a domain, i.e. a way how to view or model a domain, and of the specification of this conceptualization, e.g. a formal description.

In addition, both of the conceptualization and specification are influenced by a modeling method (e.g. frames and slots or some description logic). This can be also considered as a part of ontology (it is sometimes called meta-ontology). At the conceptualization level, we decide which objects and relations among them will be included in the ontology and also to which level of details.

At the specification level, we formally specify the conceptualization, usually in some formal language. The formalisms used can range from a simple glossary of simple terms, through an informal definition in a natural language, a formal is-a relation, a formal description of frames and properties, value restrictions, to general logical constraints. It is clear that a less formal ontology is much simpler to be developed and that a more formal ontology usually enables an easier reuse and sharing, particularly in an automatic way.

The ontology defines how to model the state of affairs in a domain together with restrictions to be considered. Ontology should capture knowledge that is not changing, while the particular state of affairs is captured in a knowledge base.

Ontologies are developed and used because they enable among others:

- to share knowledge – by sharing the understanding of the structure of information shared among software agents and people
- to reuse knowledge – ontology can be reused for other systems operating in a similar domain
- to make assumptions about a domain explicit – e.g. for easier communication

Ontologies should be well designed and also well defined. By a good design we mean that they should adequately capture the modeled domain, be understandable for a human user and provide good support for machine processing. By a good definition we mean not only the syntax, but also the semantics. The formal semantics is important if we want to introduce an automated reasoning over ontologies. Such reasoning enables to support the ontology design (such as consistency checking or supporting more authors developing one ontology), integrating and sharing ontologies automatically, determining and establishing relationships among ontologies etc.

3 Formal Concept Analysis

Formal Concept Analysis (FCA) is a theory of data analysis which identifies conceptual structures among data sets [3][1]. These structures are graphically represented as conceptual lattices, allowing the analysis of complex structures and the discovery of dependencies within the data. Formal Concept Analysis is a conceptual clustering technique with well developed mathematical foundations and was successfully used to a wide range of application in medicine, psychology, libraries, software engineering and ecology, and to a variety of methods for data analysis, information retrieval, and knowledge discovery in databases.

FCA arose twenty years ago as a theory for the formalization of the concept of "concept". It is based on the philosophical understanding that a concept is constituted by two parts: its extension which consists of all objects belonging to the concept, and its intension which comprises all attributes shared by those objects. This understanding allows to derive all concepts from a given context (data table) and to introduce a subsumption hierarchy.

4 Designing Ontologies using Formal Concept Analysis

Currently the ontology design is usually started with designing the hierarchy of relevant concepts. The taxonomical relationship, called is-a relation or subsumption relation between concepts, is perceived as the base of any ontology. This view is in our opinion influenced mainly by the procedure of designing object oriented or frame-based systems.

As a typical frame-based system we can mention Open Knowledge Base Connectivity [2], which is an API for accessing and modifying knowledge bases that are expressed in a frame-based manner. OKBC can be mapped to the object oriented languages, so that classes in programming languages can be built on the underlying ontology and be used for exchanging information. A typical design approach for modeling of the object oriented systems is the Universal Modeling Language (UML). While object oriented systems are not primarily intended for knowledge representation, the approaches are similar to the frame-based systems and the same problems are occurring here.

In these systems, the design typically starts with designing the hierarchy of classes or frames. To the existing hierarchy of classes or frames one adds attributes or properties. These attributes or properties are then inherited along the subsumption (is-a) relation. This procedure leads to several problems:

- tendency to create hierarchies of objects with no clear distinctions – for modeling of the domain, many objects are introduced, that are organized in the taxonomical ordering, but that have no other differentiating attributes; this leads to problems in knowledge sharing
- it is not easy to change frames and their slots (or classes and their attributes) once they are defined (however, so called refactoring is currently offered in the object-oriented languages as a possible solution to this problem)

To avoid these problems, and also to bring other advantages, we propose another method for constructing ontologies. The main characteristics of this method are:

- concepts are described by properties
- the properties determine the hierarchy of concepts; in other words the hierarchy is not being defined explicitly by designer
- when the properties of different concepts are the same, then the concepts are the same as well

The procedure of designing an ontology supported by a tool that uses FCA can be then described as outlined in the Figure 1. An important advantage of this process is that it can be used in a collaborative environment, with more ontology designers working on one ontology. Anyone can suggest changes to the ontology as described above, and the administrator of the ontology decides what changes to include.

1. Start with empty set of concepts and properties
2. Add concepts and properties as needed to the concept table
3. The lattice of the concepts with their properties is visualized using FCA – this enables designer(s) to see the ontology or its parts in a visual way
4. Based on the visualization, a designer can modify the ontology as follows:
    a. "Direct" editing (as directly required by ontology usage)
        i. Add or remove concept
        ii. Add or remove property
        iii. Assign a property to concept or remove a property from concept
    b. Editing as suggested by the ontology design tool
        i. When two concepts fall into one place when visualized using FCA, they should be either merged to one, or a distinction should be added (in a form of property that one concept has and the other one does not have)
        ii. The FCA can generate concepts that are formed by properties and are super-concepts of defined concepts, but are not explicitly mentioned in the concept table; this suggests that this concept can be created (ontology designer can just add a concept name upon the suggestion, the properties of the new concept are obvious from the generated lattice)
5. This process is repeated until ontology designer is satisfied

Fig. 1. Outline of the algorithm for designing ontologies using formal concept analysis.

5 Example Ontology Design

In this section, we will illustrate the process in detail on the example of designing the ontology of water geographical objects using the procedure described above. We start with the following initial objects and attributes: *lake* and *river* as objects and *flowing* and *stagnant* as attribute. Using these objects, the context cross table and the Hasse

diagram are illustrated in the first step in the figure 2 (the diagrams are generated using ToscanaJ tool [8]).

	Flowing	stagnant
lake		X
river	X	

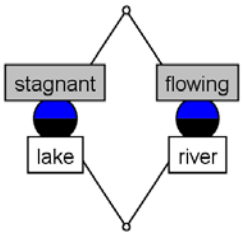

Fig. 2. Concept lattice from initial objects and attributes.

Two new concepts appeared in the visualization – the top and the bottom of the lattice. The top correspond to the concept of everything and the bottom corresponds to the concept of nothing (i.e. contradiction of attributes). So, the tool that helps with the design would ask whether there is an object that is *stagnant* and *flowing*. The user would confirm that there is no such object, because these two attributes are inverse one.

However, from the intended usage of the ontology we discover, that there is a necessity to introduce object *pond*, which can be described by the current set of attributes by *stagnant*. Following the procedure above, the context cross table and the Hasse diagram are illustrated in the figure 3.

	flowing	stagnant
lake		X
river	X	
pond		X

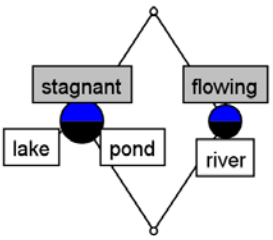

Fig. 3. Added pond object – new attribute is suggested to distinguish between lake and pond.

After visualization, we easily see that the objects *lake* and *pond* form one concept, which essentially means that they are exactly the same. It would have no sense to have two names for one concept, so as suggested by the tool following the procedure described above, we have to think about an attribute that would distinguish between these two objects. Such attribute can for example indicate whether the object is *natural* or *artificial*. After introducing these two opposite attributes, the situation looks is illustrated in the figure 4.

	flowing	stagnant	natural	artificial
Lake		X	X	
river	X		X	
pond		X		X

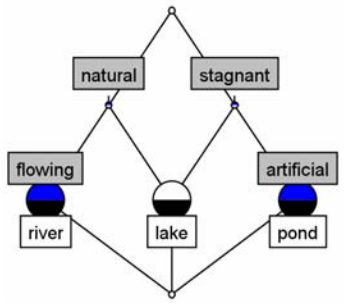

Fig. 4. A situation where new concepts can be suggested.

From this situation, we see that some new concepts can be introduced. As in the situation above, there are no objects that would be both *natural* and *artificial*, and there are no objects that would be both *flowing* and *stagnant*. However, there exist objects that would be both *flowing* and *artificial – canal* and *ditch*.

	flowing	stagnant	natural	artificial
lake		X	X	
river	X		X	
pond		X		X
canal	X			X
ditch	X			X

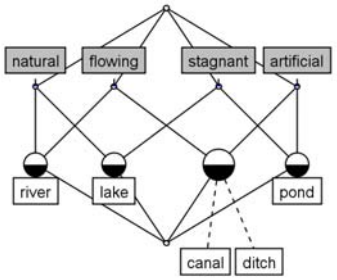

Fig. 5. Need to distinguish between two concepts.

In the need of distinguishing between *canal* and *ditch*, as suggested by the procedure above, we introduce attributes that say how large the object is – i.e. *large* and *small*. After introducing these attributes, the situation would look like as in the figure 6.

As illustrated above, there are several new concepts that can be modelled using the existing attributes, but that do not have explicit name yet – such as a concept that is *flowing* and *small*, but that is also *natural*. An object with these properties is *brook*.

In a similar way, we will add other new objects – *slough* and *basin*. The ontology after these steps is illustrated in the figure 7.

As we see from the visualization, there are a lot of new concepts, such as the one that is natural and flowing, but has no other properties. We can also eliminate the opposite attributes and visualize the lattice only for properties *large*, *natural*, and *flowing* – see figure 8. From this we can see that no other concepts using the attributes we have would make any sense, so we stop our ontology design here.

	flowing	stagnant	natural	artificial	large	small
lake		X	X		X	
river	X		X		X	
pond		X		X	X	
canal	X			X	X	
ditch	X			X		X

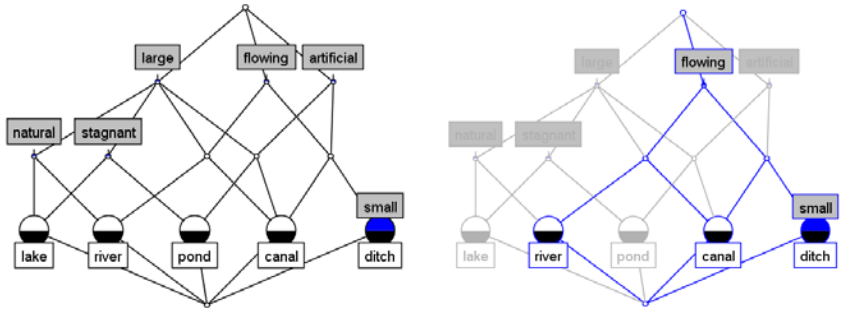

Fig. 6. Focusing on a part of the concept lattice – flowing, small objects.

	flowing	stagnant	natural	artificial	large	small
lake		X	X		X	
river	X		X		X	
pond		X		X	X	
canal	X			X	X	
ditch	X			X		X
brooks	X		X			X
slough		X	X			X
basin		X		X		X

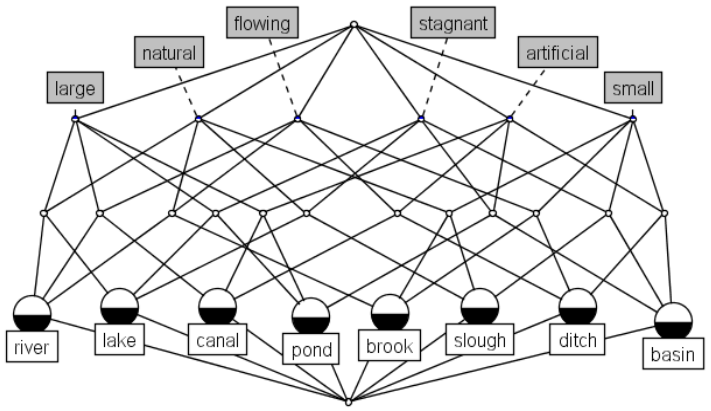

Fig. 7. Final ontology after adding all objects and properties.

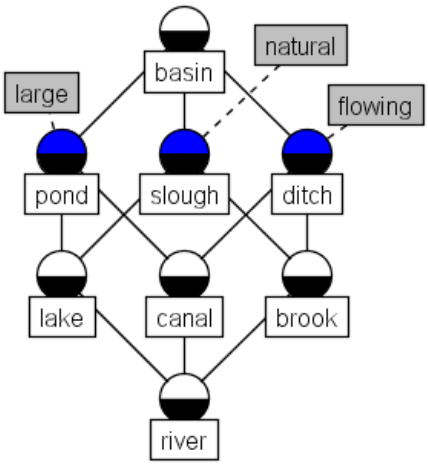

Fig. 8. Final ontology with eliminated opposite attributes.

6 Converting Ontology to Other Formalisms

The ontology designed in the previous section can be converted to any other formalism [5][2] that is used for ontology modeling. The result from the previous section can be interpreted in several ways. The intended usage determines what conversion will be actually used. Some of the possibilities are follow (this is not exhaustive list):

1. There are classes of things that are *large*, *natural*, etc., and there are instances of these classes that are *lake*, *river*, etc.
2. There are classes of things that are *large*, *natural*, etc., and there are subclasses of these classes that are *lake*, *river*, etc.
3. There are classes of things that are *lake*, *river*, etc., and properties *large*, *natural*, etc. The domain of the properties is determined by the FCA-generated lattice.
4. In the case of inverse properties, these can be directly modeled in an ontology. This means that any of the previous conversions can be easily enhanced by stating inverse properties.

Note that some of these approaches are not strictly different. For example, in description logic, the instances are often modeled as subclasses, which is only a trick for more efficient reasoning. The meaning is different, but when properly interpreted, the result after reasoning is the same.

7 Conclusion

We have presented a method for designing ontologies using a formal concept analysis and illustrated this method on an example. This method relies only on objects (or classes) and their properties, and allows to discover potential new objects and properties. Both existing and new suggested entities can be automatically shown in a visual way. A tool can guide ontology design in the described way.

We believe that this approach leads to better ontologies that are more suitable for knowledge sharing than pure taxonomies. This approach can be also used to re-engineer existing ontologies to enhance and validate them.

8 Acknowledgement

This research was supported in part by GACR grant 201/03/1318.

References

1. Beneš, M., Snášel, V.: Deducing Design Class Hierarchy from Object Properties. ISM'2002. Rožnov pod Radhoštěm. Czech Republic. 2002
2. Chaudhri, A.F.V., Fikes, R., Karp, P., Rice, J.: OKBC: A Programmatic Foundation for Knowledge Base Interoperability. Proceedings of AAAI-98. 1998
3. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundation. Berlin: Springer Verlag. 1999
4. Gruber, T.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition. 5. 1993
5. Harmelen, F.van, Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Stein, L.: Web Ontology Language (OWL) Reference. 2003. http://www.w3.org/TR/owl-ref/
6. Obitko, M.: Ontologies - Description and Applications. Report No. GL 126/01. Gerstner Laboratory for Intelligent Decision Making and Control Series of Research Reports. 2001. http://cyber.felk.cvut.cz/gerstner/reports/GL126.pdf
7. Obitko, M., Smid, J., Snášel, V.: Using Easel Types for Document Management and Retrieval. PSMP3 Workshop within AIA2004. IASTED/ActaPress. 2004
8. ToscanaJ website. http://toscanaj.sourceforge.net/