Creation and Merging of Ontology Top-Levels

Bernhard Ganter,[1] Gerd Stumme[2]

[1] Institute for Algebra, TU Dresden, D–01062 Dresden, Germany;
ganter@math.tu-dresden.de
[2] Institute for Applied Informatics and Formal Description Methods (AIFB),
University of Karlsruhe, D–76128 Karlsruhe, Germany;
www.aifb.uni-karlsruhe.de/WBS/gst

Abstract. We provide a new method for systematically structuring the top-down level of ontologies. It is based on an interactive, top–down knowledge acquisition process, which assures that the knowledge engineer considers all possible cases while avoiding redundant acquisition. The method is suited especially for creating/merging the top part(s) of the ontologies, where high accuracy is required, and for supporting the merging of two (or more) ontologies on that level.

1 Introduction

Ontologies have been established as a means for conceptually structuring domains of interest and are increasingly used for knowledge sharing. Efficient support in the creation, maintenance and interoperability of ontologies is an essential aspect for their success. Manual ontology engineering using conventional editing tools without support is difficult, labor intensive and error prone. Therefore, several systems and frameworks for supporting the knowledge engineer in the ontology engineering task have been proposed (see Section 7). These approaches often rely on syntactic and semantic matching heuristics which are derived from the behavior of ontology engineers when confronted with the task of creating and manipulating ontologies, i. e. human behaviour is simulated. Although some of them locally use different kinds of logics for comparisons, they do not provide a formal guarantee that the knowledge engineer has considered all relevant aspects in the acquisition phase.

We propose a new method, called OntEx (Ontology Exploration), for supporting the two tasks of creating and merging ontologies. It relies on the knowledge acquisition technique of Attribute Exploration [5] as developed in the mathematical framework of Formal Concept Analysis [21, 7]. OntEx guarantees that the knowledge engineer considers all relevant combinations of concepts for the creation/merging process but avoids redundant acquisition.

The first task we address is the *creation of a new ontology.* When ontologies have to be created from scratch, the user needs some guidance how to start. Especially the very first decisions have strong impact on the result, as they determine the overall structure of the ontology. Creation of ontologies from scratch is usually performed top–down. First the most general concepts of the ontology are selected. More specific concepts are then added by classifying them in the already present structure. OntEx supports one part of this creation process, namely the structuring of the conceptual hierarchy on the top–level concepts, and the creation of new concepts on the next level by providing suggestions based on an interactive exploration of the existing structure.

The second task we address is *ontology merging*. With the growing usage of ontologies, the problem of overlapping ontologies occurs more often and becomes critical. It is impossible in practice to provide a single ontology satisfying all users with regard to coverage, precision, actuality, and individualization. Hence the balance between the two conflicting objectives of providing a common knowledge core on the one hand and sufficient influence of the different users on its structure and content on the other hand has to be maintained. A solution to this problem is to provide multiple ontologies. But when the need for communication arises, the different ontologies have to be made compatible. Compatibility can be obtained by *merging* the ontologies into a unique one. Merging two ontologies means creating a new ontology in a semi-automatic manner by merging concepts of the source ontologies. OntEx provides an interactive knowledge acquisition technique for merging of the top–level concepts, where design decisions have the most impact on the overall structure of the target ontology.

The use of OntEx guarantees that the knowledge engineer considers all relevant possibilities both for the creation and for the merging task. However, this guarantee is paid with a certain workload for the knowledge engineer, making it applicable only to relatively small parts of the ontologies at hand. Therefore we propose a two–step, top–down approach. The first step aims at reliably creating/merging the top–part(s) of the ontologies with high accuracy, using OntEx. In the second step, any heuristics–based approach can be used for creating the remainder of the target ontology with fewer user interaction. This two–step approach allows for high accuracy for the design of the top–level ontology, which has large impact on the global structure of the resulting ontology, as it is more difficult to modify in a later phase than local decisions on a lower level of detail. On the other hand, it restricts the comparatively high workload on the user to the first, critical phase.

In this paper, we restrict ourselves to the construction of the concept hierarchy. The extension of this approach to relations (based on [23]) is planned for the next future. Our approach allows to make use of any background knowledge encoded in propositional logic; especially of axioms which come along with source ontologies that are to be merged.

This paper is organized as follows. In Section 2, we briefly introduce some basic definitions concentrating on a formal definition of what an ontology is and recall the basics of Formal Concept Analysis. In Section 3, the approach is sketched. Technical details are given in Section 4. It is illustrated in Section 5 by an example. Section 6 discusses how the approach is adopted to the task of ontology merging, and illustrates it by an example. In Section 7, related work is discussed. Section 8 summarizes the paper and concludes with an outlook on future work.

2 Basic Notions

In this section, we briefly introduce some basic definitions. We thereby concentrate on a formal definition of ontologies and recall the basics of Formal Concept Analysis.

2.1 Ontologies

There is no common formal definition of what an ontology is. However, most approaches share a few core items: concepts, a hierarchical IS-A-relation, and further relations. For sake of generality, we do not discuss more specific features like constraints, functions, or axioms here. We follow the definition provided in [19]:

Definition: A *(core) ontology* is a tuple $\mathcal{O} := (C, \texttt{is\_a}, R, \sigma)$, where C is a set whose elements are called *concepts*, $\texttt{is\_a}$ is a partial order on C (i. e., a binary relation $\texttt{is\_a} \subseteq C \times C$ which is reflexive, transitive, and anti-symmetric), R is a set whose elements are called *relation names* (or *relations* for short), and $\sigma: R \to C^+$ is a function which assigns to each relation name its arity.

As said above, the definition considers the core elements of most languages for ontology representation only. It is possible to map the definition to most types of ontology representation languages.

In this paper, we allow additionally a set $\mathcal{A}$ of propositional logic axioms describing dependencies between the concepts.

2.2 Formal Concept Analysis

We recall the basics of Formal Concept Analysis (FCA) as far as they are needed for this paper. A more extensive overview is given in [7]. To allow a mathematical description of concepts as being composed of extensions and intensions, FCA starts with a *formal context*:

Definition: A *formal context* is a triple $\mathbb{K} := (G, M, I)$, where G is a set of *objects*, M is a set of *attributes*, and I is a binary relation between G and M (i. e. $I \subseteq G \times M$). $(g, m) \in I$ is read "*object g has attribute m*".

From a formal context, (formal) concepts can be derived:

Definition: For $A \subseteq G$, we define $A^I := \{m \in M \mid \forall g \in A: (g, m) \in I\}$ and, for $B \subseteq M$, we define $B^I := \{g \in G \mid \forall m \in B: (g, m) \in I\}$.

A *formal concept* of a formal context (G, M, I) is defined as a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A^I = B$ and $B^I = A$. The sets A and B are called the *extent* and the *intent* of the formal concept (A, B). The *subconcept–superconcept relation* is formalized by

$$(A_1, B_1) \leq (A_2, B_2) :\Longleftrightarrow A_1 \subseteq A_2 \quad (\Longleftrightarrow B_1 \supseteq B_2) \ .$$

The set of all formal concepts of a context $\mathbb{K}$ together with the partial order $\leq$ is always a complete lattice,[1] called the *concept lattice* of $\mathbb{K}$ and denoted by $\underline{\mathfrak{B}}(\mathbb{K})$.

In the sequel, we will make use of the fact that a set $B \subseteq M$ of attributes is a concept intent iff $B = B^{II}$.

A possible confusion might arise from the double use of the word 'concept' in FCA and in ontologies. This comes from the fact that FCA and ontologies are two models for the concept of 'concept' which arose independently. In order to distinguish both notions, *we will always refer to the FCA concepts as 'formal concepts'. The concepts in ontologies are referred to just as 'concepts' or as 'ontology concepts'.* There is no direct counter-part of formal concepts in ontologies. Ontology concepts are best compared to FCA attributes, as both can be considered as unary predicates on the set of objects.

3 Creating Ontologies with OntEx

Our approach is based on the knowledge acquisition technique called *Attribute Exploration* [5]. For a given set of ontology concepts, it determines the lattice of all conjunctions of these concepts. In an interactive process, it asks the user questions of the kind "Is the conjunction of the concepts $c_1, c_2, \ldots,$ and c_n a subconcept of all of the concepts $c'_1, c'_2, \ldots,$ and c'_m?" with $n, m \geq 1$. The user can either accept or reject the subsumption. If he rejects, he has to provide a counter-example, i. e., a new object [or a new concept] which belongs to the extent of [is a subconcept of, resp.] all concepts $c_1, c_2, \ldots,$ and c_n but not to at least one of the concepts $c'_1, c'_2, \ldots,$ and c'_m. In this way, the list of subsumptions as well as the list of counter-examples grows iteratively, until all pairs of concepts are either in a subconcept–superconcept–relation or there is a counter-example prohibiting this.

The set of counter–examples can be empty at the beginning, or it can already contain some elements, if they are known from the beginning. The same holds for the subsumptions. If subsumptions are known from the beginning, they may be entered before starting the exploration.[2] The algorithm also allows to add further background knowledge expressible in propositional logic. A typical information is for instance that two (or more) concepts are mutually exclusive.

The OntEx approach can be split into three steps:

1. initialization of the exploration contexts,
2. the exploration process,
3. further processing.

In the initialization step, the user has to provide an initial set of concepts he considers to be relevant. How to obtain this initial set is out of the scope of this

[1] I. e., for each set of formal concepts, there exists always a greatest common subconcept and a least common superconcept.

[2] This is especially important for the merging task, where each ontology already comes along with its own subsumption hierarchy.

paper. It may be determined by other knowledge acquisition techniques as for instance described in [18]. Then one formalizes both the background knowledge (especially the known subsumptions) and the (possibly empty) set of counter–examples. The exploration process comprises the exploration dialogue with the user, consisting of questions as described above. At the end of this process, the lattice of all conjunctions of the input concepts is determined. It contains all input concepts and some new concepts constructed during the process. In the third phase, the user can modify the resulting hierarchy using any ontology editor, eventually supported by some heuristic approach as described in Section 7.

4 The OntEx Method

In this section, we discuss in detail the three steps for creating the concept hierarchy of the (top–level of the) ontology in detail. An example is provided in the next section.

In order to simplify notations, we will not distinguish between objects (instances) and sub-concepts as counter-examples in the sequel. In practice, of course this distinction will be tracked, and will be used as additional information for further processing the exploration results.

4.1 Initialization of the Exploration Contexts

In the initialization step, we formalize both the background knowledge (especially the known subsumptions) and the (possibly empty) set of counter–examples in form of formal contexts. We assume that the user has already fixed some initial set C of ontology concepts (without necessarily any hierarchy information on it) which he considers important. Further concepts may be added at later iterations of the process.

In order to simplify notations, we will not encode the background knowledge in form of axioms, but rather as a list of potential examples.[3] This list contains thus all combinations of attributes which are not excluded by the background knowledge. It is stored as a context $\mathbb{P} := (G, C, I)$, called *frame context*, where C is the initial set of concepts. The set G consists at the first stage of dummy concepts. They will be replaced by concrete concept names (or will be deleted) during the exploration. There is one $g \in G$ for each attribute combination $B \subseteq C$ with $\{g\}^I = B$ unless the combination is known to be impossible according to the eventually given background knowledge.[4]

[3] Here and in the sequel we will not address performance issues – which are of course important – but rather try to keep the explanation as simple as possible. The performance of attribute exploration with background knowledge is analyzed in detail in [6]. There it is shown that the fact that the exploration is restricted to implicational logic (whereas the background knowledge may be given by arbitrary propositional formulae) makes the approach computationally feasible.

[4] The fact that in the worst case $\mathrm{card}(G) = 2^{\mathrm{card}(C)}$ holds, indicates the importance of formalizing as much background knowledge as possible in order to reduce the size of $\mathbb{P}$.

If we know already of some instances of ontology concepts at the very beginning, then we code this information in two more contexts, $\mathbb{O}^{+} := (O, C, I^{+})$ and $\mathbb{O}^{?} := (O, C, I^{?})$ with $O \subseteq G$. $\mathbb{O}^{+}$ encodes which of the objects are known to belong to which concepts, i.e., $(o, c) \in I^{+}$ iff we know that object o belongs to concept c. $\mathbb{O}^{?}$ encodes which of the objects are not known not to belong to certain concepts, i.e., $(o, c) \in I^{?}$ iff we cannot exclude that object o belongs to concept c. In this encoding, we have $I^{+} \subseteq I \cap (O \times C) \subseteq I^{?}$.

4.2 The Exploration Process

In the exploration process, the set G of possible objects will decrease, and the set O of counter–examples will increase, based on the answers given by the user, until the implicational theory of the contexts is equal. Then the exploration process ends; and the resulting implicational theory is the one of the target ontology. In particular, it describes the subsumption hierarchy of the target ontology.

At each step in the interactive process, OntEx checks if there is any set $X \subseteq C$ of concepts with $X = X^{II} \neq X^{??}$.[5] If so, then such a set is chosen and the user is asked if the implication $\bigwedge X \to \bigwedge(X^{??})$ is universally valid for the ontology (see details below). If no such X exists then the implicational theory of the ontology is completely determined, and there are no more "open questions". The concept lattice of $\mathbb{P}$ will contain the full concept order. But even more: we then know all cases where a conjunction of concepts subsumes another concept. The conjunctions can be understood as *implicit* encodings of new concepts, and will be used later as seeds for new concepts for the target ontology.

When there still exists a set $X \subseteq C$ of concepts with $X = X^{II} \neq X^{??}$, then 'the next such set' is chosen[6] and the user is asked if the conjunction of all concepts in X is a subconcept of all concepts in $X^{??}$ (i.e., if $\bigwedge X \to \bigwedge(X^{??})$ is universally valid). Then the user has to provide a counter-example or has to accept the implication.[7]

Each single input to the exploration process can be given in form of a clause $\bigwedge A \to \bigvee N$ (with $A, N \subseteq C$), which is marked either as *valid* or as *non-valid*. The input of a "valid" clause (i.e., the partial acceptance of a subsumption suggested by the system) will cause modifications of $\mathbb{P}$, $\mathbb{O}^{+}$, and of $\mathbb{O}^{?}$. From $\mathbb{P}$ all objects not consistent with $\bigwedge A \to \bigvee N$ are removed, and the partial description of the objects $o \in O$, encoded by $\mathbb{O}^{+}$ and $\mathbb{O}^{?}$, is updated according to $\bigwedge A \to \bigvee N$.

A clause $\bigwedge A \to \bigvee N$ which is marked as "non–valid" is interpreted as a partial description of a counter–example, i.e. of a new object $o \in O$ with $\{o\}^{+} := A$ and $\{o\}^{?} := C \setminus N$. The consistency of the description of the given counter–example with the background knowledge and the results already obtained has to

[5] We write $X^{?}$ and X^{+} instead of $X^{I^{?}}$ and $X^{I^{+}}$, resp. (see Section 2.2).
[6] In [4] (see also [7]), an efficient strategy is described.
[7] Depending on the underlying ontology language, the user can be supported in this task by some additional inference mechanism, for instance a description logics inference engine (see [1]).

be checked. This can be done using standard methods of propositional logics and will not be described in this paper. If the user says that the counter–example is a concept, then it may additionally be added to the set C. In this case, it will be considered during the subsequent exploration phase. This optional extension of the set C of concepts provides higher accuracy, as also all combinations with this new concept are considered, but it extends the duration of the knkowledge acquisition.

When the user has given his answer, and the exploration contexts are modified as described, the 'next' set $X \subseteq C$ with $X = X^{II} \neq X^{??}$ (according to the modified contexts) is determined[8] and the next question is asked to the user.

In the worst case (i. e., when all subsumptions are rejected without exception) and with the worst answering strategy, the number of questions is exponential in the cardinality of the initial set of concepts. In praxis, however, this worst–case complexity is far from being reached, since there are numerous dependencies between the concepts. On the other hand, the underlying theory guarantees that the number of accepted subsumptions is minimal, and can thus not be outperformed by any other technique.

4.3 Further Processing

Having finished the exploration process, we have now completely determined the subsumption order on the initial set of concepts, and we know all constraints between these concepts which can be described in implicational logic. This includes in particular the information which combinations of concepts any instances can have, and which combinations are excluded. This information is added to the set of axioms of the ontology.

At this point the high accuracy creation of the target ontology is finished. Now heuristics–based approaches may take over. They may be selected out of the list of tools as described in Section 7.

An additional aspect resulting from our approach is that the possible combinations (conjunctions) of initial concepts are seeds for new concepts. A name for a newly generated concept can be derived from its minimal generators (i. e., the minimal subsets of the set of initial concepts whose conjunctions are equal to the new concept) as described in detail in [19]. The knowledge engineer may take over some or all of the new concepts. Taking over all of them has the advantage that the resulting ontology is a lattice which allows for computation with the concepts (i. e., for any given set of concepts, the computation of its unique least common superconcept and its unique greatest common subconcept; and the computation of implications (functional dependencies) between the concepts). On the other hand, the resulting ontology may become too large. It depends on the individual application how this trade–off will be resolved.

[8] The technique described in [4, 7] guarantees that the order on the sets $X \subseteq C$ is compatible with the modification of the contexts.

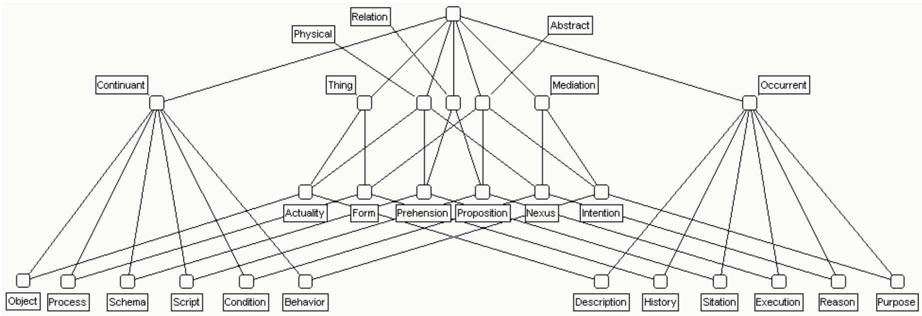

Fig. 1. Sowa's top–level ontology. The bottom concept is not shown for sake of readability.

5 An Example Application

In [16], J. F. Sowa elaborated a top–level ontology, based on a careful study of the ontology work of the philosophers Heraclitus, Aristotle, Kant, Peirce, Husserl, Whitehead, and Heidegger. Using the three distinctions 'Physical — Abstract' of Heraclitus, 'Thing — Relation — Mediation' of Peirce, and 'Continuant — Occurrent' of Whitehead, Sowa established the top–level ontology shown in Figure 1. In this section, we show how OntEx could have helped Sowa in this approach.[9]

We initialize the exploration by letting $C := \{$Physical, Abstract, Thing, Relation, Mediation, Continuant, Occurrent$\}$, $G := \mathfrak{P}(C)$ (i. e., we do not exclude any combinations of concepts at the beginning), and $I := \{(g,c) \in G \times C \mid c \in g\}$. Finally, we let $O := \emptyset$ and $I^+ := I^? := \emptyset$. The contexts are now determined, and the exploration dialogue can begin:

The empty set is the first set satisfying $X = X^{II} \neq X^{??}$. For $X = \emptyset$, we have $X^{II} = \emptyset$ and $X^{??} = C$.

Q: Is the conjunction of no concepts at all (i. e., the top concept) a subconcept of all of the concepts Physical, Abstract, Thing, Relation, Mediation, Continuant, and Occurrent?

A: No. A counter–example is the new concept Object, which is subconcept of the top concept, but beside that only of the concepts Continuant, Thing, and Physical.

The set O is extended by Object,[10] and I^+ and $I^?$ are extended to $\{$(Object, Continuant),(Object,Thing),(Object,Physical)$\}$. One might also add the new concept Object to the set C, but we do not make use of this option here.[11]

[9] In [22], the same example has been used to explain 'Simply Implicational Theories', a formalism for supporting empicrical theory building within the framework of Formal Concept Analysis.

[10] In the set G, Object is identified with $\{$Continuant, Thing, Physical$\} \in G$.

[11] Choosing this option would lead to some more questions, and would in the end result in the additional observation that Object is in fact the conjunction (i. e.,

We choose again $X := \emptyset$, since now $X^{II} = \emptyset$ and $X^{??}$ ={CONTINUANT, THING, PHYSICAL}.

Q: Is the conjunction of no concepts (i. e., the top concept) a subconcept of all of the concepts PHYSICAL, THING, and CONTINUANT?

A: No. A counter–example is the new concept PURPOSE, which is subconcept of the top concept, but beside that only of the concepts OCCURRENT, MEDIATION, and ABSTRACT (and is hence not subconcept of any of the concepts mentioned).

The set O is extended by PURPOSE, and I^+ and $I^?$ are extended by the three tuples (PURPOSE,OCCURRENT), (PURPOSE,MEDIATION), and (PURPOSE,ABSTRACT).

Now we have $X^{II} = \emptyset = X^{??}$ for $X = \emptyset$, hence the next set has to be chosen. With the strategy described in [5], we obtain X ={OCCURRENT} (with X^{II} ={OCCURRENT} and $X^{??}$ ={OCCURRENT, MEDIATION, ABSTRACT}).

Q: Is OCCURRENT a subconcept of all of the concepts MEDIATION and ABSTRACT?

A: No. A counter–example is the new concept DESCRIPTION, which is subconcept of OCCURRENT, but beside that only of the concepts THING and PHYSICAL (and is hence not subconcept of any of the concepts mentioned).

Again O, I^+, and $I^?$ are extended according to the rules. This way, the exploration continues. The first question we pass along which will be accepted is the following:

Q: Is the conjunction of RELATION and MEDIATION a subconcept of all other concepts?

A: Yes (because these two concepts are considered to be disjoint and their conjunction is thus equal to the "absurd" bottom element).

Now the set G is decreased by subtracting all its elements g with {RELATION, MEDIATION$\subseteq g$; and from the relation I are subtracted all tuples having these sets as first component.

This way, the exploration takes on, until no sets X remain fulfilling the condition $X = X^{II} \neq X^{??}$. After 23 questions in total, we obtain the concept hierarchy shown in Figure 2.

In the first level, we see the concepts we started with. In the second and third level, we see the concepts which were given as counter-examples. We can for instance see that BEHAVIOR is a subconcept of the three concepts CONTINUANT, ABSTRACT, and MEDIATION. (If we had decided to add BEHAVIOR to the set C during the exploration, we would additionally know that it is indeed equal to their greatest common subsumer.) The ten concepts without label are

the greatest common subconcept) of the three concepts CONTINUANT, THING, and PHYSICAL, and not just an arbitrary subconcept of them.

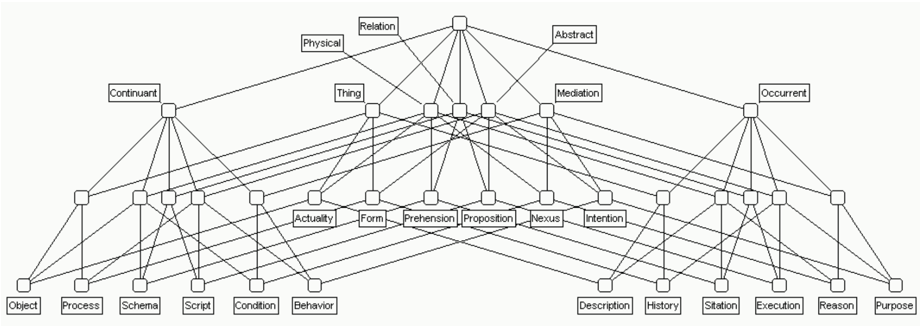

Fig. 2. The result of ONTEX based on the categories of Heraclitus, Peirce, and Whitehead. The bottom concept is not shown for sake of readability.

formal concepts in the sense of FCA. If we discard them, then we obtain exactly Sowa's top–level ontology in Figure 1. However, they can also be used as further concepts of the top–level ontology. It is then up to the knowledge engineer to provide names for them. If all these concepts are accepted, we have the additional advantage that one can compute with the concepts, since the resulting concept hierarchy is a lattice: each pair of concepts has meet (i. e., a unique greatest common subconcept) and join (i. e., a unique least common superconcept).

In the diagram, we can see that none of the concepts we started with is subsumed by any other one. By studying the diagram closer, we discover that all possible combinations of concepts out of each of the sets of categories of Heraclitus, Peirce, and Whitehead are realized as a new concept. Hence the sets of categories of the three philosophers are truly orthogonal to each other. According to Sowa (see Figure 1), though, the combination of one category of Whitehead with one category of either Heraclitus or Peirce (e. g., OCCURRENT and MEDIATION) is not a concept of its own. This is of course a philosophical question, and cannot be solved by our algorithm. However, ONTEX makes these potential concepts explicit and accessible to discussion.

6 Ontology Merging with OntEx

In this section, we discuss the use of ONTEX for ontology merging. The main difference to the task of creating an ontology from scratch is the initialization phase. The input of the merging process are two[12] ontologies $\mathcal{O}_1 := (C_1, \texttt{is\_a}_1, R_1, \sigma_1)$ and $\mathcal{O}_2 := (C_2, \texttt{is\_a}_2, R_2, \sigma_2)$, eventually together with two sets $\mathcal{A}_1$ and $\mathcal{A}_2$ of propositional logics axioms. The output is then a target ontology $\mathcal{O}_T := (C_T, \texttt{is\_a}_T, R_T, \sigma_T)$ together with a set of axioms $\mathcal{A}_T$. As said above, we will not deal with the exploration of the relations here. Their treatment by ONTEX is subject to further research. We simply let $R_T := R_1 \cup R_2$.

[12] The approach is applicable to more than two ontologies simultanuously.

We illustrate the approach with a small example. It consists of the two mini-ontologies shown in Figure 3. The two ontologies come along with two sets of axioms, namely $\mathcal{A}_1 := \{\emptyset \to \textsc{Root1}, \neg(\textsc{Area1} \wedge \textsc{Hotel1})\}$ and $\mathcal{A}_2 := \{\emptyset \to \textsc{Root2}, \neg(\textsc{Region2} \wedge \textsc{Accommodation2})\}$.

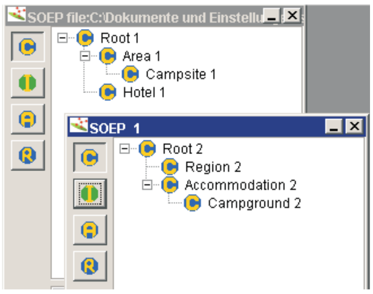

Fig. 3. The two example source ontologies.

6.1 Initialization of the Exploration Contexts

We have to initialize the frame context $\mathbb{P} := (G, C, I)$ and the two contexts $\mathbb{O}^+ := (O, C, I^+)$ and $\mathbb{O}^? := (O, C, I^?)$ according to the given knowledge. Let first of all $C := C_1 \dot\cup C_2$, $O := \emptyset$, and $I^+ := I^? := \emptyset$. The set G first contains $2^{\mathrm{card}(C)}$ objects such that, for each $B \subseteq C$, there exists exactly one $g \in G$ with $\{g\}' = B$.

Then the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ of axioms are turned into a set of clauses marked as valid resp. invalid (using standard logic techniques). For each pair $(c, c') \in (C_1 \times C_1) \cup (C_2 \times C_2)$ of concepts in the two ontologies where c is an immediate subconcept of c' (i.e., a lower cover in the `is_a` hierarchy), we add $c \to c'$ as valid and $c' \to c$ as invalid clause. The contexts $\mathbb{P}$, $\mathbb{O}^+$ and $\mathbb{O}^?$ are then modified as described in the fourth paragraph of Section 4.2. If the two ontologies (or one of them) come along with instances, then they are added to the set O, and I^+ and $I^?$ are updated as described for new counter–examples in Section 4.2. With the three contexts initialized this way, the exploration starts.

6.2 The Exploration Process

The exploration dialogue follows the same rules as described in Section 4.2. For our example, this means that the first question is as follows:

Q: Is $\textsc{Hotel1}$ a subconcept of all of $\textsc{Region2}$, $\textsc{Campground2}$ and $\textsc{Accommodation2}$?
A: No. A counter–example is 'Raffles Hotel'.

The sets O and I^+ are extended accordingly: The set O is extended by the object 'Raffles Hotel', and the relations I^+ and $I^?$ by $\{$'Raffles Hotel'$\} \times \{$ROOT1, HOTEL1, ROOT2, ACCOMMODATION2, $\}$.

Q: Is HOTEL1 a subconcept of ACCOMMODATION2?
A: Yes.

Hence the set G of the frame context $\mathbb{P}$ is pruned by deleting all objects $g \in G$ where $\{g\}'$ contradicts HOTEL1 $\rightarrow$ ACCOMMODATION2 .

The exploration continues until all open questions are solved. Finally, the set $\mathcal{A}_T$ of axioms of the target ontology is obtained (without any further user interaction) by $\mathcal{A}_T := \mathcal{A}_1 \cup \mathcal{A}_2 \cup \{c_1 \leftrightarrow c_2 \mid c_1 \in C_1$ and $c_2 \in C_2$ are merged in the target ontology$\}$. In our example, we have

$$\mathcal{A}_T := \mathcal{A}_1 \cup \mathcal{A}_2 \cup \{\textsc{Root1} \leftrightarrow \textsc{Root2}, \textsc{Campsite1} \leftrightarrow \textsc{Campground2}\} \ .$$

The $\mathtt{is\_a}_T$ hierarchy is shown in Figure 4, together with the first counter–example acquired during the exploration process. In total, there are two more counter–examples: 'Yosemite Camping', being a counter–example to both CAMPSITE1 $\rightarrow$ REGION2 and CAMPGROUND2 $\rightarrow$ HOTEL1; and 'Tuscany', being a counter–example to REGION2 $\rightarrow$ HOTEL1.

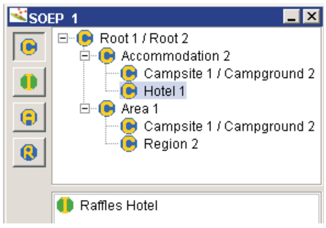

Fig. 4. The target ontology obtained by ONTEX.

6.3 Further Processing

For this third step, all remarks made in Section 4.3 are valid. Additionally, ONTEX may have determined pairs of concepts of the source ontologies which are merged in the target ontology (as ROOT1/ROOT2 and CAMPSITE1/CAMPGROUND2 in our example). In the preprocessing step, these concepts can serve as starting point for further merging of concepts and relations by addressing next concepts which are close to them in the source ontologies. This approach is for instance realized in Prompt [14] (but with a simpler, more straightforward way of obtaining the initially merged concepts).

7 Related Work

A first approach for supporting the merging of ontologies is described in [8]. There, several heuristics are described for identifying corresponding concepts in different ontologies, e. g. comparing the names and the natural language definitions of two concepts, and checking the closeness of two concepts in the concept hierarchy.

The OntoMorph system [3] offers two kinds of mechanisms for translating and merging ontologies: syntactic rewriting supports the translation between two different knowledge representation languages, semantic rewriting offers means for inference-based transformations. It explicitly allows to violate the preservation of semantics in trade-off for a more flexible transformation mechanism.

In [12] the Chimaera system is described. It provides support for merging of ontological terms from different sources, for checking the coverage and correctness of ontologies and for maintaining ontologies over time. Chimaera offers a broad collection of functions, but the underlying assumptions about structural properties of the ontologies at hand are not made explicit.

Protégé-2000 [13] is a knowledge acquisition tool and ontology editor which can be used for creating ontologies. Prompt [14] is an algorithm for ontology merging and alignment embedded in Protégé 2000. It starts with the identification of matching class names. Based on this initial step an iterative approach is carried out for performing automatic updates, finding resulting conflicts, and making suggestions to remove these conflicts.

OilEd [2] is an editor for ontologies based on the description language FaCT. OntoEdit [17] is an ontology editor based on Frame Logic. These tools make use of inferences for checking the consistency and for deriving new facts from the knowledge base.

About merging, there is also much related work in the database community, especially in the area of federated database systems. The work closest to our approach is described in [15]. There, Formal Concept Analysis is applied to a related problem, namely database schema integration.

FCA–MERGE [19] is a technique for merging ontologies based on Formal Concept Analysis. It offers a structural description of the overall merging process with an underlying mathematical framework. It relies on the existence of instances annotated in both ontologies, and provides alternatively a way to use documents as such instances. Concepts are suggested to be merged merged iff they have the same extent.

Efficient support for creating ontologies from scratch is topic of current research. In [9–11], for instance, text mining techniques have been discussed for supporting the user in creating ontologies.

Most of the tools described above offer extensive functionalities, often based on syntactic and semantic matching heuristics, which are derived from the behaviour of ontology engineers when confronted with the task of creating/merging ontologies. OntoMorph, Chimarea, OilEd, and OntoEdit use a (description) logics based approach that influences the creation and merging process locally, e. g.

checking subsumption relationships between terms and checking for inconsistencies. However, none of these approaches offers a guarantee that all relevant relationships have been considered for modeling during the acquisition phase.

8 Conclusion and Future Work

In this paper, we presented OntEx (Ontology Exploration), a technique for creating/merging ontologies with high accuracy. The technique is suited especially for small parts of the ontologies — especially its top parts — where high quality results are required. Its results can be used as starting point for heuristics–based approaches for dealing with the remainder of the ontologies.

In the paper, we described the three steps of the technique: the initialization of the exploration contexts, the exploration process, and the further processing. We also discussed how the approach can be used for merging (the top levels of) ontologies. The paper described the underlying assumptions and discussed the overall methodology.

Future work includes the closer integration of the method in the KAON ontology engineering environment.[13] It is also planned to extend this approach to the creation/merging of the ontology relations by combining the Attribute Exploration technique with Rule exploration [23], which allows for first order literals instead of binary attributes. Here, decidability will be an important issue, hinting at the use of some description logic.

References

1. F. Baader: Computing a minimal representation of the subsumption lattice of all conjunctions of concept defined in a terminology. In: G. Ellis, R. A. Levinson, A. Fall, V. Dahl (eds.): *Proc. Intl. KRUSE Symposium*, August 11–13, 1995, UCSC, Santa Cruz 1995, 168–178
2. S. Bechhofer, I. Horrocks, C. Goble, R. Stevens: OilEd: A Reason-able Ontology Editor for the Semantic Web, In: *KI–2001: Advances in Artificial Intelligence*, LNAI **2174**, Springer, Heidelberg 2001, 396–408
3. H. Chalupsky: OntoMorph: A translation system for symbolic knowledge. *Proc. KR '00*, Breckenridge, CO, USA,471–482.
4. B. Ganter: Algorithmen zur Formalen Begriffsanalyse. In: B. Ganter, R. Wille, K. E. Wolff (eds.): *Beiträge zur Formalen Begriffsanalyse*, B. I.–Wissenschaftsverlag, Mannheim 1987, 241–254
5. B. Ganter: Attribute Exploration with Background Knowledge. *TCS* **217**(2), 1999, 215–233
6. B. Ganter, R. Krauße: *Pseudo models and propositional Horn inference.* Preprint MATH-AL-15-1999, TU Dresden 1999
7. B. Ganter, R. Wille: *Formal Concept Analysis: mathematical foundations.* Springer, Berlin–Heidelberg 1999

[13] kaon.semanticweb.org

8. E. Hovy: Combining and standardizing large-scale, practical ontologies for machine translation and other uses. *Proc. 1st Intl. Conf. on Language Resources and Evaluation*, Granada, Spain, May 1998.
9. A. Mädche: *Ontology Learning for the Semantic Web*. PhD thesis, Universität Karlsruhe. Kluwer, Dordrecht 2002
10. A. Mädche, S. Staab: Semi-automatic engineering of ontologies from Text. *Proc. 12th Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE'2000)*
11. A. Mädche, S. Staab: Discovering conceptual relations from text. *Proc. 14th European Conference on Artificial Intelligence (ECAI 2000)*, IOS Press, Amsterdam, 2000
12. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder: An environment for merging and testing large Ontologies. *Proc. KR '00*, 483–493.
13. N. F. Noy, R. Fergerson, M. Musen: The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility, *Proc. EKAW 2000*, LNCS **1937**, Springer, Heidelberg 2000, 17–32
14. N. F. Noy, M. A. Musen: PROMPT: algorithm and tool for automated ontology merging and alignment. *Proc. AAAI '00*, 450–455
15. I. Schmitt, G. Saake: Merging inheritance hierarchies for database integration. *Proc. CoopIS'98*, IEEE Computer Science Press, 322–331.
16. J. F. Sowa: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publ., Pacific Grove, CA, 2000
17. S. Staab, A. Mädche: Ontology engineering beyond the modeling of concepts and relations. *Proc. ECAI'2000 workshop on application of ontologies and problem-solving methods*, IOS Press, Amsterdam 2000
18. S. Staab, H.-P. Schnurr, R. Studer, Y. Sure: Knowledge Processes and Ontologies. *IEEE Intelligent Systems* **16**(1), 2001
19. G. Stumme, A. Mädche: FCA–Merge: Bottom-Up Merging of Ontologies. *Proc. 17th Intl. Conf. on Artificial Intelligence (IJCAI '01)*. Seattle, WA, USA, 2001, 225–230
20. Y. Sure, C. Boyens: OntoKick – Ignition for Ontologies. Poster Session at: *WI–IF 2001 — 5th International Conference WirtschaftsInformatik*, Sep. 19–21, 2001, Augsburg, Germany
21. R. Wille: Restructuring lattice theory: an approach based on hierarchies of concepts. In: I. Rival (ed.): *Ordered sets*. Reidel, Dordrecht 1982, 445–470
22. S. Strahringer, R. Wille, U. Wille: Mathematical Support for Empirical Theory Building. In: H. Delugach, G. Stumme (Eds.): *Conceptual Structures: Broadening the Base*. Proc. ICCS '01. LNAI **2120**, Springer, Heidelberg 2001, 169–186
23. M. Zickwolff: *Rule Exploration: First Order Logic in Formal Concept Analysis*. PhD thesis, TH Darmstadt 1991