

Ontologies

In this chapter, we introduce our formal ontology model. The model presented will provide a basis for the formalization of ontology learning tasks in Chapter 3 as well as for the evaluation measures used throughout the remainder of the book.

The term *ontology* comes from the Greek *ontologia* and means “talking” (-logia) about “being” (ὄν / onto-). Ontology is a philosophical discipline which can be described as the *science of existence* or the *study of being*. Platon (427 - 347 BC) was one of the first philosophers to explicitly mention the *world of ideas or forms* in contrast to the real or observed objects, which according to his view are only imperfect realizations (or *shadows*) of the ideas (compare [Annas, 1981]). In fact, Platon raised ideas, forms or abstractions to entities which one can talk about, thus laying the foundations for ontology. Later his student Aristotle (384 - 322 BC) shaped the logical background of ontologies and introduced notions such as *category*, *subsumption* as well as the superconcept/subconcept distinction which he actually referred to as *genus* and *subspecies*. With *differentiae* he referred to characteristics which distinguish different objects of one genus and allow to formally classify them into different categories, thus leading to subspecies. This is the principle on which the modern notions of ontological concept and inheritance are based upon. In fact, Aristotle can be regarded as the founder of *taxonomy*, i.e. the science of classifying things. Aristotle’s ideas represent the foundation for object-oriented systems as used today. Furthermore, he introduced a number of inference rules, called *sylogisms*, such as those used in modern logic-based reasoning systems [Sowa, 2000a].

In modern computer science parlance, one does not talk anymore about ‘ontology’ as the science of existence, but of ‘ontologies’ as formal specifications of a conceptualization in the sense of Gruber [Gruber, 1993]. So, whereas ‘ontology’ was originally a science, ‘ontologies’ have received the status of resources representing the conceptual model underlying a certain domain, describing it in a declarative fashion and thus cleanly separating it from procedural aspects.

Whereas the number of applications for ontologies in computer science is steadily growing, the necessity for a clear and formal definition of an ontology arises at the same time. In the past, there have been many proposals for an ontology language with a well-defined syntax and formal semantics, especially in the context of the Semantic Web, such as OIL [Horrocks et al., 2000], RDFS [Brickley and Guha, 2002] or OWL [Bechhofer et al., 2004]. In the context of this book, we will however stick to a more mathematical definition of ontologies in line with Stumme et al. [Stumme et al., 2003]. Our definitions are to a great extent borrowed from there. However, we take the freedom to modify the definitions for our purposes. Furthermore, we illustrate the definition with a running example.

Definition 1 (Ontology) *An ontology is a structure*

$$\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, \mathcal{A}, \sigma_A, \mathcal{T})$$

consisting of

- *four disjoint sets C , R , \mathcal{A} and \mathcal{T} whose elements are called concept identifiers, relation identifiers, attribute identifiers and data types, respectively,*
- *a semi-upper lattice \leq_C on C with top element root_C , called concept hierarchy or taxonomy,*
- *a function $\sigma_R: R \rightarrow C^+$ called relation signature,*
- *a partial order \leq_R on R , called relation hierarchy, where $r_1 \leq_R r_2$ implies $|\sigma_R(r_1)| = |\sigma_R(r_2)|$ and $\pi_i(\sigma_R(r_1)) \leq_C \pi_i(\sigma_R(r_2))$, for each $1 \leq i \leq |\sigma_R(r_1)|$, and*
- *a function $\sigma_A: \mathcal{A} \rightarrow C \times \mathcal{T}$, called attribute signature,*
- *a set \mathcal{T} of datatypes such as strings, integers, etc.*

Hereby, $\pi_i(t)$ is the i -th component of tuple t . In some cases, when it is clear from the context whether we are referring to a relation or an attribute, we will simply use σ .

Further, a semi-upper lattice \leq fulfills the following conditions:

$$\forall x \ x \leq x \text{ (reflexive)} \tag{2.1}$$

$$\forall x \forall y \ (x \leq y \wedge y \leq x \rightarrow x = y) \text{ (anti-symmetric)} \tag{2.2}$$

$$\forall x \forall y \forall z \ (x \leq y \wedge y \leq z \rightarrow x \leq z) \text{ (transitive)} \tag{2.3}$$

$$\forall x \ x \leq \text{top} \text{ (top element)} \tag{2.4}$$

$$\forall x \forall y \exists z \ (z \geq x \wedge z \geq y \wedge \forall w \ (w \geq x \wedge w \geq y \rightarrow w \geq z)) \tag{2.5}$$

(supremum)

So every two elements have a unique most specific supremum. In the context of ontologies, we will refer to this element as the *least common subsumer*. It is obviously defined as follows:

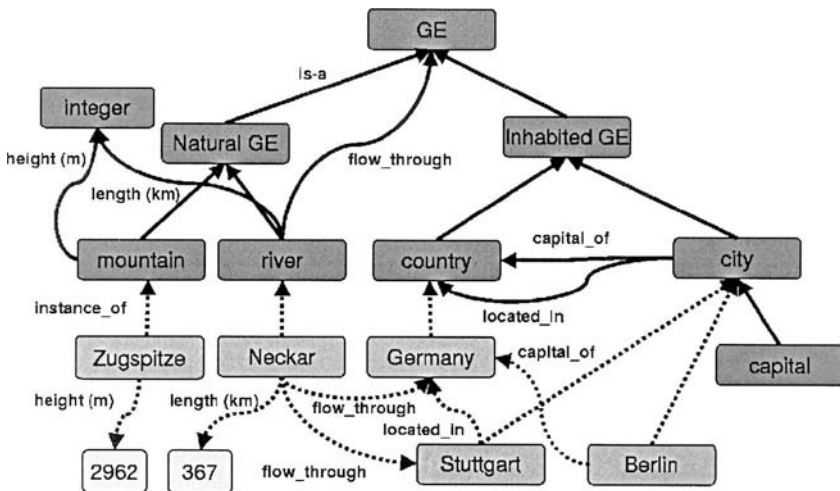


Fig. 2.1. Example ontology

$$lcs(a, b) := z \text{ such that } z \geq a \wedge z \geq b \text{ and } \forall w (w \geq a \wedge w \geq b \rightarrow w \geq z) \quad (2.6)$$

Often we will call concept identifiers and relation identifiers just *concepts* and *relations*, respectively, for the sake of simplicity. For binary relations, we define their *domain* and their *range* as follows:

Definition 2 (Domain and Range) For a relation $r \in R$ with $|\sigma(r)| = 2$, we define its domain and range by $\text{dom}(r) := \pi_1(\sigma(r))$ and $\text{range}(r) := \pi_2(\sigma(r))$.

If $c_1 <_C c_2$, for $c_1, c_2 \in C$, then c_1 is a *subconcept* of c_2 , and c_2 is a *superconcept* of c_1 . If $r_1 <_R r_2$, for $r_1, r_2 \in R$, then r_1 is a *subrelation* of r_2 , and r_2 is a *superrelation* of r_1 .

If $c_1 <_C c_2$ and there is no $c_3 \in C$ with $c_1 <_C c_3 <_C c_2$, then c_1 is a *direct subconcept* of c_2 , and c_2 is a *direct superconcept* of c_1 . We note this by $c_1 \prec c_2$. *Direct superrelations* and *direct subrelations* are defined analogously.

Let us illustrate all the above definitions on the basis of a simple example ontology graphically depicted in Figure 2.1. The set C of concepts is $C := \{\text{GE}, \text{Natural GE}, \text{Inhabited GE}, \text{mountain}, \text{river}, \text{country}, \text{city}, \text{capital}\}$, where GE stands for *geographical entity*. The set R of relations is: $R := \{\text{located_in}, \text{flow_through}, \text{capital_of}\}$. Further, we have two attributes,

i.e. $A := \{\text{length (km)}, \text{height (m)}\}$. According to the direct superconcept relation we have, from left to right: $\text{mountain} \prec \text{Natural GE}$, $\text{river} \prec \text{Natural GE}$, $\text{Natural GE} \prec \text{GE}$, $\text{country} \prec \text{Inhabited GE}$, $\text{city} \prec \text{Inhabited GE}$, $\text{capital} \prec \text{city}$ and $\text{Inhabited GE} \prec \text{GE}$. The partial order \leq_C is then $\leq_C := \prec \cup \{(\text{mountain, GE}), (\text{river, GE}), (\text{country, GE}), (\text{city, GE}), (\text{capital, Inhabited GE}), (\text{capital, GE})\}$.

In our example, the top element of the concept upper semi-lattice is $\text{root}_C := \text{GE}$. Further, $\text{lcs}(\text{country, city})$ is for example Inhabited GE , whereas $\text{lcs}(\text{city, capital})$ is city and $\text{lcs}(\text{mountain, city})$ is GE .

For the relations and attributes in the example ontology we have the following signatures:

$\sigma_R(\text{flow_through}) = (\text{river, GE})$
 $\sigma_R(\text{capital_of}) = (\text{city, country})$
 $\sigma_R(\text{located_in}) = (\text{city, country})$
 $\sigma_A(\text{length (km)}) = (\text{river, integer})$
 $\sigma_A(\text{height (m)}) = (\text{mountain, integer})$

The relation hierarchy could further include $\text{capital_of} \leq_R \text{located_in}$, i.e. if x is capital of y , then x is also located in y .

Having defined the basic elements of a core ontology, we now define an axiom system for it. Though we are not directly concerned with learning axioms, we introduce an axiom system for the sake of completeness.

Definition 3 (\mathcal{L} -Axiom System) *Let \mathcal{L} be a logical language. A \mathcal{L} -axiom system for an ontology $\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, A, \sigma_A, \mathcal{T})$ is a triple*

$$S := (AS, \alpha, \mathcal{L})$$

where

- AS is a set whose elements are called axiom schemata and
- $\alpha: AS \rightarrow AS_{\mathcal{L}}$ is a mapping from AS to axiom schemata defined over \mathcal{L} .

An ontology with an \mathcal{L} -axiom system is a pair

$$(\mathcal{O}, S)$$

where \mathcal{O} is an ontology and S is an \mathcal{L} -axiom system for \mathcal{O} .

We will formalize these axiom schemata using the untyped lambda calculus (compare [Barendregt, 1984]) originally introduced by Church [Church, 1936]. The lambda calculus essentially provides a means to describe arbitrary unnamed functions. A lambda expression consists of a variable which we abstract over - the argument of the function - and which is bound by the λ operator. A function $f(x) = x^2$ can thus be written in the lambda calculus notation as $\lambda x.x^2$, where the dot (.) separates the lambda operator from the actual body of the function. In what follows, we will regard the standard lambda calculus notation as equivalent to the *uncurried* notation in which lists of λ -bound

variables are used. Thus, $\lambda x. (\lambda y. (x + y))$ will be written in the more handy form: ' $\lambda x, y. x + y$ ', omitting the parenthesis by assuming that the λ -operator binds the variables in the list until the end of the whole expression.

For example, one axiom schema could be $\lambda P, Q. disjoint(P, Q)$ which is mapped by α to a first-order logic schema as

$$\lambda P, Q. \forall x (P(x) \rightarrow \neg Q(x)).$$

$\alpha(disjoint)(river)(mountain)$ would thus yield:

$$\forall x (river(x) \rightarrow \neg mountain(x)).$$

The obvious benefit of such an \mathcal{L} -axiom system is that by being independent of some concrete knowledge representation formalism, the axioms formulated can be translated into a variety of different languages. This is important for ontology learning as the statements learned from textual data have in fact an intuitive interpretation independent of any knowledge representation formalism. The learned statements can then get assigned a specific interpretation with respect to a concrete KR formalism via the α mapping. Axiom schemata capture frequently occurring patterns used in ontology engineering (compare [Staab et al., 2001]). In addition to instantiations of these axiom schemata, other general axioms have to be added to the logical theory. The difference between axiom schemata and general axioms is thus only a pragmatic one, i.e. it depends on the fact whether a type of general axiom occurs often enough to deserve the status of an axiom schema. For example, we will assume the following two axioms as being part of our logical theory:

$$\begin{aligned} &\forall x (country(x) \rightarrow \exists y capital_of(y, x) \wedge \forall z (capital_of(z, x) \rightarrow z = y)) \\ &\forall x (capital(x) \leftrightarrow \exists y capital_of(x, y) \wedge country(y)) \end{aligned}$$

The first axiom states that every country has a unique capital, while the second defines the concept capital as equivalent to saying that there is a country which stands in a *capital_of* relation with the corresponding city. Depending on the view adopted and if axioms as the above occur frequently, one could introduce the following axiom schema:

$$\lambda C_1, C_2, R. C_1 = \exists R. C_2$$

which would be mapped to the following first-order axiom schema:

$$\lambda C_1, C_2, R. \forall x (C_1(x) \leftrightarrow \exists y \wedge R(x, y) \wedge C_2(y))$$

The instantiation $\lambda C_1, C_2, R. C_1 = \exists R. C_2(\text{capital})(\text{country})(\text{capital_of})$ would then be mapped to the following first-order formula:

$$\forall x (capital(x) \leftrightarrow \exists y \wedge capital_of(x, y) \wedge country(y))$$

The crucial question here certainly is whether the corresponding axiom occurs frequently enough to be lifted to the status of an axiom schema.

In what follows, we also define what a lexicon for an ontology is:

Definition 4 (Lexicon) *A lexicon for an ontology*

$$\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, \mathcal{A}, \sigma_A, T)$$

is a structure

$$Lex := (S_C, S_R, S_A, Ref_C, Ref_R, Ref_A)$$

consisting of

- *three sets S_C , S_R and S_A whose elements are called signs for concepts, relations and attributes, respectively,*
- *a relation $Ref_C \subseteq S_C \times C$ called lexical reference for concepts,*
- *a relation $Ref_R \subseteq S_R \times R$ called lexical reference for relations, and*
- *a relation $Ref_A \subseteq S_A \times \mathcal{A}$ called lexical reference for attributes.*

Based on Ref_C , we define, for $s \in S_C$,

$$Ref_C(s) := \{c \in C \mid (s, c) \in Ref_C\}$$

and, for $c \in C$,

$$Ref_C^{-1}(c) := \{s \in S_C \mid (s, c) \in Ref_C\}.$$

Ref_R and Ref_R^{-1} as well as Ref_A and Ref_A^{-1} are defined analogously.

An ontology with lexicon is a pair

$$(\mathcal{O}, Lex),$$

where \mathcal{O} is an ontology and Lex is a lexicon for \mathcal{O} .

For our example ontology, we could for instance specify that both *nation* and *country* refer to the concept *country*, i.e. $Ref_C^{-1}(\text{country}) = \{\text{nation}, \text{country}\}$.

It is important to mention that the above definition accommodates a great variety of lexical structures to which concepts and relations can refer, depending how the sets S_C , S_R and S_A are defined. In fact, they could merely contain labels, i.e. plain strings for the concepts and relations as typically assumed, but also highly structured objects (compare [Buitelaar et al., 2006]).

Whereas ontologies formally specify the conceptualization of a domain, the extensional part is provided by a knowledge base which contains assertions about instances of the concepts and relations.

Definition 5 (Knowledge Base (KB)) A knowledge base for an ontology $\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, \mathcal{A}, \sigma_A, \mathcal{T})$ is a structure

$$KB := (I, \iota_C, \iota_R, \iota_A)$$

consisting of

- a set I whose elements are called instance identifiers (or instances or objects for short),
- a function $\iota_C: C \rightarrow 2^I$ called concept instantiation,
- a function $\iota_R: R \rightarrow 2^{I^+}$ with $\iota_R(r) \subseteq \prod_{1 \leq i \leq |\sigma(r)|} \iota_C(\pi_i(\sigma(r)))$, for all $r \in R$. The function ι_R is called relation instantiation, and
- a function $\iota_A: \mathcal{A} \rightarrow I \times \bigcup_{t \in \mathcal{T}} \llbracket t \rrbracket$ with $\iota_A(a) \subseteq \iota_C(\pi_1(\sigma(a))) \times \llbracket \pi_2(\sigma(a)) \rrbracket$, where $\llbracket t \rrbracket$ are the values of datatype $t \in \mathcal{T}$. The function ι_A is called attribute instantiation.

In our example ontology, we have for instance: $I := \{\text{Zugspitze, Neckar, Germany, Stuttgart, Berlin}\}$. Further, we have the following instantiation relations:

$$\begin{aligned} \iota_C(\text{mountain}) &:= \{\text{Zugspitze}\} \\ \iota_C(\text{river}) &:= \{\text{Neckar}\} \\ \iota_C(\text{country}) &:= \{\text{Germany}\} \\ \iota_C(\text{city}) &:= \{\text{Stuttgart, Berlin}\} \\ \iota_R(\text{flow_through}) &:= \{(\text{Neckar, Germany}), (\text{Neckar, Stuttgart})\} \\ \iota_R(\text{located_in}) &:= \{(\text{Stuttgart, Germany})\} \\ \iota_R(\text{capital_of}) &:= \{(\text{Berlin, Germany})\} \\ \iota_A(\text{length (km)}) &:= \{(\text{Neckar, 367})\} \\ \iota_A(\text{height (m)}) &:= \{(\text{Zugspitze, 2962})\} \end{aligned}$$

As for concepts and relations, we also provide names for instances.

Definition 6 (Instance Lexicon) An instance lexicon for a knowledge base $KB := (I, \iota_C, \iota_R, \iota_A)$ is a pair

$$IL := (S_I, R_I)$$

consisting of

- a set S_I whose elements are called signs for instances,
- a relation $R_I \subseteq S_I \times I$ called lexical reference for instances.

A knowledge base with lexicon is a pair

$$(KB, IL)$$

where KB is a knowledge base and IL is an instance lexicon for KB .

When a knowledge base is given, we can derive the extensions of the concepts and relations of the ontology based on the concept instantiation and the relation instantiation.

Definition 7 (Extension) Let $KB := (I, \iota_C, \iota_R, \iota_A)$ be a knowledge base for an ontology $\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, \mathcal{A}, \sigma_A, \mathcal{T})$. The extension $\llbracket c \rrbracket_{KB} \subseteq I$ of a concept $c \in C$ is recursively defined by the following rules:

- $\llbracket c \rrbracket_{KB} \leftarrow \iota_C(c)$
- $\llbracket c \rrbracket_{KB} \leftarrow \llbracket c \rrbracket_{KB} \cup \llbracket c' \rrbracket_{KB}$, for $c' <_C c$.
- instantiations of axiom schemata in S (if \mathcal{O} is an ontology with \mathcal{L} -axioms),
- other general axioms contained in the logical theory.

The extension $\llbracket r \rrbracket_{KB} \subseteq I^+$ of a relation $r \in R$ is recursively defined by the following rules:

- $\llbracket r \rrbracket_{KB} \leftarrow \iota_R(r)$
- $\llbracket r \rrbracket_{KB} \leftarrow \llbracket r \rrbracket_{KB} \cup \llbracket r' \rrbracket_{KB}$, for $r' <_R r$.
- instantiations of axiom schemata in S (if \mathcal{O} is an ontology with \mathcal{L} -axioms),
- other general axioms contained in the logical theory.

The extension $\llbracket a \rrbracket_{KB} \subseteq I \times \llbracket \mathcal{T} \rrbracket$ of an attribute $a \in \mathcal{A}$ is defined as:

- $\llbracket a \rrbracket_{KB} \leftarrow \iota_A(a)$
- general axioms contained in the logical theory.

If the reference to the knowledge base is clear from the context, we also write $\llbracket c \rrbracket$, $\llbracket r \rrbracket$ and $\llbracket a \rrbracket$ instead of $\llbracket c \rrbracket_{KB}$, $\llbracket r \rrbracket_{KB}$ and $\llbracket a \rrbracket_{KB}$. Given our example, we get in particular (taking into account the relation hierarchy and our general axioms defining capitals and their relation to countries):

```

[[mountain]] := {Zugspitze}
[[river]] := {Neckar}
[[country]] := {Germany}
[[city]] := {Stuttgart, Berlin}
[[capital]] := {Berlin}
[[Natural GE]] := {Zugspitze, Neckar}
[[Inhabited GE]] := {Germany, Berlin, Stuttgart}
[[GE]] := {Germany, Berlin, Stuttgart, Zugspitze, Neckar}
[[flow_through]] := {(Neckar, Germany), (Neckar, Stuttgart)}
[[located_in]] := {(Stuttgart, Germany), (Berlin, Germany)}
[[capital_of]] := {(Berlin, Germany)}

```

Finally, what is missing is a definition of the intension of a certain concept or relation. We extend the definitions of Stumme et al. [Stumme et al., 2003] to also accommodate the intension of concepts and relations as follows:

Definition 8 (Intension) A structure

$$\mathcal{I} := (\mathcal{L}_I, i_C, i_R, i_A)$$

is called the intension of an ontology $\mathcal{O} := (C, \leq_C, R, \sigma_R, \leq_R, \mathcal{A}, \sigma_A, \mathcal{T})$ and consists of:

- a language \mathcal{L}_I capturing intensions of concepts, relations and attributes, respectively,
- three mappings i_C , i_R and i_A with $i_C : C \rightarrow \mathcal{L}_I$, $i_R : R \rightarrow \mathcal{L}_I$ and $i_A : A \rightarrow \mathcal{L}_I$, mapping concepts, relations and attributes to their corresponding intensions.

We interpret the *intension* as a non-extensional definition of a certain concept or relation. The above definition also accommodates different languages for expressing the intension of concepts and relations. The *intension*, for example, could be represented through *differentiae* in the sense of Aristotle explaining why a certain concept is different from others and thus merits a status on its own. In this line, the language could consist of sets of attributes describing a concept in line with the theory of Formal Concept Analysis (see Section 4.2). However, the language could consist of strings describing the intuitive meaning of a concept in natural language such as done with the *glosses* of the WordNet lexical resource [Fellbaum, 1998] (compare Section 4.1.8). In this line, in our example the intension for capital could be $i_C(\text{capital}) := \text{'town or city that is the center of government of a country, state or province'}$. Having outlined our formal ontology model, the next chapter introduces the core topic of the book, i.e. ontology learning from text.

Ontology Learning and Population from Text
Algorithms, Evaluation and Applications

Cimiano, P.

2006, XXVIII, 347 p., Hardcover

ISBN: 978-0-387-30632-2