

Definitions

Any good research needs a strong foundation. This chapter lays these foundations for the book. One characteristic of original research is the lack of a standardized vocabulary. Therefore, we will first explain the terminology of ontology and alignment. Then the concept of similarity and its meaning for ontologies will be described, on both an abstract and a concrete level with examples. Only with a thorough understanding of the main expressions, it is possible to follow the ideas of the succeeding chapters.

2.1 Ontology

In the last years, ontologies have increasingly been used in computer science, but also in biology and medicine, or knowledge management. Accompanying this development, the definitions of ontology have varied considerably. As *ontology* is one of the key terms, this section will define its further usage here.

2.1.1 Ontology Definition

In philosophy, ontology is the theory of “*the nature of being or the kinds of existents*” (mer, 2006). The Greek philosophers Socrates and Aristotle were the first developing the foundations of ontology. Socrates introduced the notion of abstract ideas, a hierarchy among them, and class-instance relations. Aristotle added logical associations. The result is a well-structured model, which is capable of describing the real world. Still, it is not trivial to include all the extensive and complex relations of our environment. Looking at ontology from a different point of view, from mathematics, we perceive a complex directed graph, which represents knowledge about the world. This model is extended with logical axioms to allow for inferencing. In modern history, first papers resuming the philosophical discipline ontology were published around 1960 (Strawson and Bubner, 1975).

Artificial Intelligence and web researchers have adopted the term “ontology” for their own needs. Currently there are different definitions in the literature of what an ontology should be. Some of them are discussed in Guarino (1997), the most prominent, focused on here, being “*An ontology is an explicit specification of a conceptualization.*” (Gruber, 1995). A conceptualization refers to an abstract model of some phenomenon in the world by identifying the relevant concept of that phenomenon (Studer et al., 1998). Explicit means that the types of concepts used and the constraints on their use are explicitly defined. This definition is often extended by three additional conditions: “*An ontology is an explicit, formal specification of a shared conceptualization of a domain of interest.*” where formal refers to the fact that the ontology should be machine-readable (which excludes, for instance, natural language). Shared reflects the notion that an ontology captures consensual knowledge, i.e., it is not private to an individual. Shared does not necessarily mean globally shared, but accepted by a group. Thus, the integration problem addressed in this work therefore stays unsolved by this definition. Ontology alignment remains necessary. Finally, the reference to a domain of interest indicates that for domain ontologies one is not interested in modeling the whole world, but rather in modeling just those parts of a certain domain relevant to the task.

Common to all these definitions is their high level of generalization, which is far from a precise mathematical definition. The reason for this is that the definition should cover all different kinds of ontologies, and should not be related to a particular method of knowledge representation (van Heijst et al., 1997). However, to study structural aspects, we have to commit ourselves to one specific ontology model and to a precise, detailed definition. This section presents the definition of this key term ontology that has been developed in the knowledge management group at the Institute AIFB at the University of Karlsruhe. The definition adheres to the Karlsruhe Ontology Model as expressed in Stumme et al. (2003).

Definition 2.1 (Core Ontology). *A core ontology is a structure*

$$S := (C, \leq_C, R, \sigma, \leq_R)$$

consisting of

- *two disjoint sets C and R whose elements are called concept identifiers and relation identifiers (or concepts and relations for short),*
- *a partial order \leq_C on C , called concept hierarchy or taxonomy,*
- *a function $\sigma: R \rightarrow C \times C$ called signature,¹ where $\sigma(r) = \langle \text{dom}(r), \text{ran}(r) \rangle$ with $r \in R$, domain $\text{dom}(r)$, and range $\text{ran}(r)$,*
- *a partial order \leq_R on R , called relation hierarchy, where $r_1 \leq_R r_2$ iff $\text{dom}(r_1) \leq_C \text{dom}(r_2)$ and $\text{ran}(r_1) \leq_C \text{ran}(r_2)$.*

¹ In contrast to some other definitions, we here actually restrict the model to binary relations.

For simplification, datatypes such as integers or strings are treated as special kinds of concepts, $D \subset C$. Further, we say, 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 $c_1 <_C c_2$ 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 denote this by $c_1 \prec c_2$. *Super- and subrelations* as well as their direct counterparts are defined analogously. The core ontology is often also referenced to as schema.

Relationships between concepts and/or relations as well as constraints can be expressed within a logical language such as first-order logic or Horn-logic. We here provide a generic definition, which allows the use of different languages and logics.

Definition 2.2 (Axioms). *Let L be a logical language. An L -axiom system for a core ontology is a pair*

$$A := (AI, \alpha)$$

where

- AI is a set whose elements are called *axiom identifiers* and
- $\alpha: AI \rightarrow L$ is a mapping.

The elements of $A := \alpha(AI)$ are called *axioms*. S is considered to be part of the language L .

The core ontologies formalize the intentional aspects of a domain. The extensional aspects are provided by knowledge bases, which contain assertions about instances of the concepts and relations.

Definition 2.3 (Knowledge Base). *A knowledge base is a structure*

$$KB := (C, R, I, \iota_C, \iota_R)$$

consisting of

- two disjoint sets C and R as defined before,
- a set I whose elements are called *instance identifiers* (or *instances* for short),
- a function $\iota_C: C \rightarrow \mathfrak{P}(I)$ called *concept instantiation*,
- a function $\iota_R: R \rightarrow \mathfrak{P}(I^2)$ with $\iota_R(r) \subseteq \iota_C(\text{dom}(r)) \times \iota_C(\text{ran}(r))$, for all $r \in R$. The function ι_R is called *relation instantiation*.

With datatypes being concepts as stated for core ontologies, concrete values are analogously treated as instances, $V \subset I$.

We provide names for the concepts (and relations). Instead of name, we call them sign to allow for more generality.

Definition 2.4 (Lexicon). *A lexicon for an ontology is a structure*

$$Lex := (G_C, G_R, G_I, Ref_C, Ref_R, Ref_I)$$

consisting of

- three sets G_C , G_R and G_I whose elements are called *signs* for concepts, relations, and instances, resp.,
- a relation $\text{Ref}_C \subseteq G_C \times C$ called *lexical reference for concepts*, Ref_R and Ref_I analogously.

In this work, an ontology consists of a core ontology, axioms, instantiating data in the knowledge base, as well as a corresponding lexicon.

Definition 2.5 (Ontology). *An ontology O is therefore defined through the following tuple:*

$$O := (S, A, KB, Lex)$$

consisting of

- the core ontology S ,
- the L -axiom system A ,
- the knowledge base KB , and
- the lexicon Lex .

In this book we will often refer to a set of entities E . An entity $e \in E$ interpreted in an ontology O is either a concept, a relation, or an instance, i.e., $e|_O \in C \cup R \cup I$. We usually write e instead of $e|_O$ when the ontology O is clear from the context of the writing.

By enhancing these definitions with an actual ontology language, such as OWL in the next section, ontologies are given a well-defined semantics. Through axioms, it is possible to formalize a wide range of correlations. Especially the expressive semantics distinguishes ontologies from other schema such as topic maps,² XML trees (Bray et al., 2004),³ database schemas, or classification schemas, e.g., modeled through UML.⁴ The semantics of ontologies allows inferring additional knowledge. In our case, we will extensively use the defined semantics to derive alignments.

2.1.2 Semantic Web and Web Ontology Language (OWL)

The internet is a large platform for information. After its impressive development in the past decade, an incredible amount of data is now available. World knowledge seems easily reachable. Still, we are far away from the point where we can access knowledge just as easily as browsing through the web. Unfortunately, most of the information provided is meant to be human-readable only. It is in a non-standardized form and unstructured in the relation it uses. The main problem of sharing knowledge with people via computer rises from the missing capabilities of the machine to recognize the meaning of the processed information. Solving this dilemma will be a goal for the next years.

² <http://www.topicmaps.org/xtm/1.0/>

³ <http://www.w3.org/XML/>

⁴ <http://www.uml.org/>

The vision of a Semantic Web was first brought up by Berners-Lee et al. (2001). *“The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users.”* The web will become machine-readable and processable based on ontologies improving its effectiveness by magnitudes. In a more recent interview (Berners-Lee, 2005), he added: *“The Semantic Web is designed to smoothly interconnect personal information management, enterprise application integration, and the global sharing of commercial, scientific, and cultural data. We are talking about data here, not human documents.”* As the prospective killer application he sees company intranets where sharing with semantics begins in a smaller controlled environment. Just as the internet started in closed environments, the Semantic Web is also expected to take this path.

An approach to overcome the barrier of machines not recognizing meaning is the application of ontologies to formalize knowledge in a machine and human readable form. This enables the user to search for information based on meaning rather than syntax. A key issue is therefore the definition of standards to represent the underlying structures, the ontologies. Each language offers different modeling primitives. RDF-Schema (Brickley and Guha, 2004),⁵ a schema language built on top of the Resource Description Framework (RDF), allows to define taxonomies and relations between concepts. The Web Ontology Language (OWL) (Smith et al., 2004; Patel-Schneider et al., 2004)⁶ is more expressive. OWL was developed keeping the primitives of description logics (Baader et al., 2003) in mind. As in this work ontology alignment is supposed to make use of expressive semantics, it will rely on OWL. Apart from RDF(S) and OWL, one should mention F-Logic as a logic-based ontology representation (Kifer et al., 1995).

OWL is currently recommended by the World Wide Web Consortium (W3C) as language for modeling ontologies for the Semantic Web. Historically, it evolved from the DAML+OIL⁷ language, which was developed by joining the European standard OIL (Ontology Inference Layer) with its American counterpart DAML (DARPA Agent Markup Language) into a unified framework. OWL is an expressive language, allowing many ontology-modeling paradigms, such as specifying hierarchical relationships between classes and restrictions and for modeling attributes and associations under a well-defined semantics. The latter allows inferring information that is not explicitly present in the ontology. Furthermore, OWL provides for the description of the state of the world by asserting information about individuals and relationships between them. There exist three sublanguages of OWL, here arranged according to their complexity: OWL Lite presents a subset that allows for easy inferring; OWL DL includes many constructs from description logics; OWL Full

⁵ <http://www.w3.org/RDF/>

⁶ <http://www.w3.org/TR/owl-features/>

⁷ <http://www.w3.org/TR/daml+oil-reference/>

finally allows any use of RDF statements, thus problems with inferencing may arise. In specific, we will focus on OWL DL. It has the following predefined constructs on concepts, relations, and instances:

- subsumption of concepts and relations;
- domains and ranges of relations;
- (in)equality definitions;
- symmetry, transitivity, reflexivity of relations;
- restrictions on relations;
- cardinality constraints;
- boolean combinations of concept expressions: union, intersection, complement;
- enumeration of instances to define a concept;
- inverse relations;
- lexical annotations such as labels or comments;
- and ontology information such as version numbers or import requests for other ontologies.

Whereas some constructs are directly reflected in the ontology definition of the previous section, others correspond to additional axioms. Currently, efforts are made to extend the ontology model with a rule language. SWRL, the Semantic Web Rule Language (Horrocks and Patel-Schneider, 2004),⁸ offers possibilities for representing complex axioms and first implementations have been provided.

2.1.3 Ontology Example

To clarify the term ontology as used here, this section contains an ontology example. We will repeatedly refer to this example throughout the work to illustrate the approaches for ontology alignment. The ontology describes the domain of automobiles as a car dealer who has modeled his stock and customer relations might have. It is a simple example, but noticeably helps to explain the basic ontology constructs.

The ontology is graphically shown in Figure 2.1. Concepts are depicted as rectangular boxes, relations as hexagons, and instances as rounded boxes. Subsumption relations are drawn as solid arrows. A relation has an incoming arrow from its domain and an outgoing arrow to its range. The instantiations of concepts and relations are depicted as dotted, arrowed lines. The example contains the six concepts object, vehicle, owner, boat, car, and speed, the two relations of belonging to somebody and speed, and the three instances Marc, Porsche KA-123, and 300 km/h. There is a subsumption relation between object, vehicle, and boat, resp. car; a vehicle is an object, a boat is a vehicle, etc. Each vehicle belongs to an owner and each car has a specific speed. On instance level, the Porsche KA-123 belongs to Marc and has the speed 300

⁸ <http://www.daml.org/rules/proposal/>

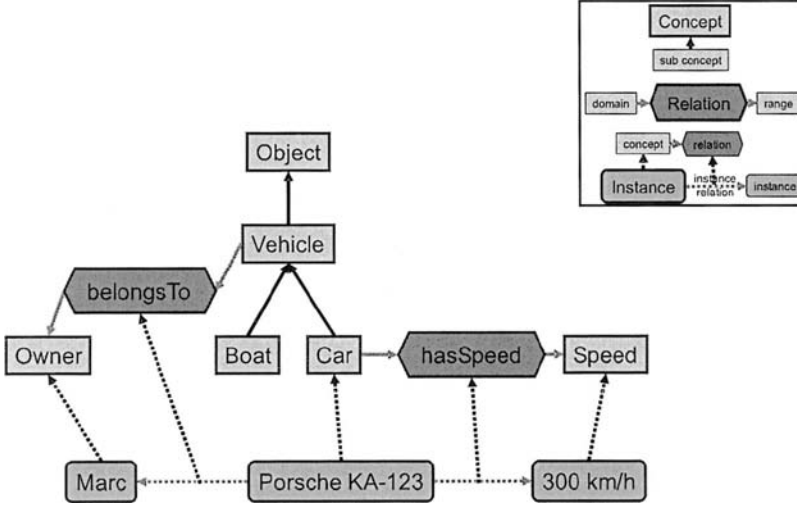


Fig. 2.1. Ontology Example

km/h. Further, the axiom that every car needs to have at least one owner is defined. Axioms are not depicted in the graph, but represented in the next paragraphs. The example is fictitious and any concurrences with the real world are purely by chance.

Formally, this ontology is defined according to $O := (S, A, KB, Lex)$. To keep the representation as short and illustrating as possible, it does not contain all constructs of the above example.

- schema $S = (C, \leq_C, R, \sigma, \leq_R) = (\{object, vehicle, owner, \dots\}, \{(vehicle, object), (boat, vehicle), \dots\}, \{belongsTo, hasSpeed\}, \{belongsTo \rightarrow (vehicle, owner), hasSpeed \rightarrow (car, speed)\}, \{\})$
- axioms⁹ $A = \{\forall x car(x) \Rightarrow \exists y belongsTo(x, y)\}$
- knowledge base $KB = (C_{KB}, R_{KB}, I, \iota_C, \iota_R) = (\{object, vehicle, owner, \dots\}, \{belongsTo, hasSpeed\}, \{Marc, PorscheKA123, 300km/h\}, \{owner \rightarrow \{Marc\}, car \rightarrow \{PorscheKA123\}, speed \rightarrow \{300km/h\}\}, \{belongsTo \rightarrow \{(PorscheKA123, Marc)\}, hasSpeed \rightarrow \{(PorscheKA123, 300km/h)\}\})$
- In the example the lexicon only contains the identifiers as lexical entries, i.e., $Lex = (\{“object”, “vehicle”, \dots\}, \dots, \{ (“object”, object), (“vehicle”, vehicle), \dots\}, \dots)$.

Finally, this ontology is expressed in OWL (Example 2.1), or more precisely the RDF/XML representation of OWL. Again each construct is only

⁹ The notation here is according to first-order logic.

mentioned once, thus the OWL representation does not include the whole example ontology. As common in XML, a namespace `auto` is added in the OWL representation. For readability, the namespaces of RDF resources are also abbreviated. After the namespace declaration a class (in our terms: concept) `auto#vehicle` is defined. This concept has the English label “*vehicle*”. It is defined to be a subconcept of `auto#object` before the class-tag is closed again. The other entities have some more tags, but are built-up accordingly. The axiom is represented as `owl:Restriction`.

Example 2.1 (Example Ontology).

```
<rdf:RDF
...
xmlns:auto="http://www.aifb.uni-karlsruhe.de/WBS/meh/auto1.owl">

<owl:Class rdf:about='auto#vehicle'>
  <rdfs:label xml:lang='en'>vehicle</rdfs:label>
  <rdfs:subClassOf rdf:resource='auto#object' />
</owl:Class>
<owl:Class rdf:about='auto#car'>
  <rdfs:label xml:lang='en'>car</rdfs:label>
  <rdfs:subClassOf rdf:resource='auto#vehicle' />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource='auto#belongsTo' />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
...
<owl:ObjectProperty rdf:about='auto#belongsTo'>
  <rdfs:label xml:lang='en'>belongs to</rdfs:label>
  <rdfs:domain rdf:resource='auto#vehicle' />
  <rdfs:range rdf:resource='auto#owner' />
</owl:ObjectProperty>
...
<auto:Owner rdf:about='auto#Marc' />
...
<auto:Car rdf:about='auto#Porsche KA-123'>
  <rdfs:label xml:lang='en'>Porsche KA-123</rdfs:label>
  <auto:belongsTo rdf:resource='auto#Marc' />
  <auto:hasSpeed rdf:resource='auto#300 km/h' />
</auto:Car>
</rdf:RDF>
```


This example concludes the definition of ontologies, our basic semantically rich structures. They will ease the task of alignment by allowing exploiting their semantics.

2.2 Ontology Alignment

The considered research field of ontology alignment and integration features a big number of terms such as alignment, mapping, mediation, merging, etc. Unfortunately, the definitions from different authors are confusing, partially inconsistent, and at times even contradicting. The goal of this section is therefore to adhere to one definition for each term. Especially *alignment*, the second key term in the title of this work, will be explained.

2.2.1 Ontology Alignment Definition

The dictionary (mer, 2006) again gives a general comprehensible sense for alignment. To align something means, “*to bring into line*”. This very brief definition already emphasizes that aligning is an activity after which the involved objects are in some mutual relation.

We here define our use of the term ontology alignment similarly to Klein (2001). Given two ontologies, aligning one ontology with another one means that for each entity (concept, relation, or instance) in the first ontology, we try to find a corresponding entity, which has the same intended meaning, in the second ontology. An alignment therefore is a one-to-one equality relation. Obviously, for some entities no corresponding entity might exist.

Definition 2.6 (Ontology Alignment). *An ontology alignment function, $align$, based on the set E of all entities $e \in E$ and based on the set of possible ontologies O is a partial function*

$$align : E \times O \times O \rightharpoonup E$$

We write $align_{O_1, O_2}(e)$ for $align(e, O_1, O_2)$. We leave out O_1, O_2 when they are evident from the context and write $align(e)$ instead. Once a (partial) alignment, $align$, between two ontologies O_1 and O_2 is established, we say entity e is aligned with entity f when $align(e) = f$. A pair of entities (e, f) that is not yet in $align$ and for which appropriate alignment criteria still need to be tested is called a candidate alignment. In this work, if not explicitly mentioned otherwise, alignment is an equality alignment.

Apart from one-to-one equality alignments, as mainly investigated here and most of the related existing work, in real world one entity often has to be aligned not only to equal entities, but based on another relation (e.g., subsumption). Further, there are complex composites such as a concatenation of terms (e.g., name equals first plus last name) or an entity with restrictions

(e.g., a sports car is a car going faster than 250 km/h). Do and Rahm (2002) and Dhamankar et al. (2004) propose approaches for identifying such alignments. As many complex alignments consist of elementary alignments, our work may be seen as basis for this. Our elementary alignments are not restricted to equality relations. We therefore extend the above basic definition of alignment by introducing a set of alignment relations M . M includes but is not restricted to identity, subsumption, instantiation, and orthogonality. A set of alignments based on taxonomical relations is presented in Giunchiglia et al. (2004). In fact, towards the end of this book in Section 10.2, we will show how our approach also fits this problem of general ontology alignment.

Definition 2.7 (General Ontology Alignment). *A general ontology alignment function, $genalign$, based on the vocabulary, E , of all terms $e \in E$, based on the set of possible ontologies, O , and based on possible alignment relations, M , is a partial function*

$$genalign : E \times O \times O \rightharpoonup E \times M$$

This last understanding is part of the alignment definition researchers have agreed on in the context of Knowledge Web (Euzenat et al., 2004),¹⁰ a European network of excellence of leading institutions on semantic technologies. Supporting the transition process of ontology technology from academia to industry is the major goal of Knowledge Web. This will be achieved through standardization, education, and integrating research. Responding to the integration task, the partners defined an alignment as a set of correspondences (i.e., quadruples): $\langle e, f, r, l \rangle$ with e and f being the two aligned entities, r representing the relation holding between them, and l (additionally) expressing the level of confidence $[0, 1]$ in the alignment statement.

2.2.2 Ontology Alignment Representation

Currently there is no generally agreed standardized format for saving ontology alignments. In this section we therefore propose the two possible representation formats that are most accepted and used in the alignment community.

The first possibility is to adhere to existing constructs, e.g., in OWL. OWL provides equality axioms for concepts, relations, and instances: `owl:equivalentClass`, `owl:equivalentProperty`, and `owl:sameAs`. It is also possible to express inequality through `owl:differentFrom`. The advantage of this representation is that OWL inference engines will automatically interpret the alignment and reason across several ontologies. The downside is the very strict equality. A confidence value cannot be interpreted accordingly. Complex alignments as mentioned before are not possible.

The second possibility is based on work of Euzenat (2004). The representation uses RDF/XML to formalize ontology alignments. After the general

¹⁰ <http://knowledgeweb.semanticweb.org/>

definition of the involved ontologies, the individual alignments are represented in cells with each cell having the attributes entity 1, entity 2, measure (the confidence), and the relation (normally '='). This representation corresponds to the Knowledge Web definition of alignment (Definition 2.7). Due to its different parameters, it can easily be used for many alignment applications. Unfortunately, it is not directly in an ontology format. Therefore, an explicit import is required to transform the alignments into a suitable format for inferencing. For this importer one also needs to define how to handle confidence values of an alignment.

Alternative representations are the MAFRA semantic bridging ontology (SBO), Contextualized OWL, the rule language SWRL, the OMWG mapping language (OML), and SKOS. An overview thereof is found in Euzenat et al. (2006).

In the later chapters of this work we will not provide alignments in either of the just introduced formats to keep the representation of alignments simple. Alignments are presented in tables with each row containing the tuple of entities and confidence $\langle e, f, l \rangle$. This tuple can be easily translated into either of the two previous formats.

2.2.3 Ontology Alignment Example

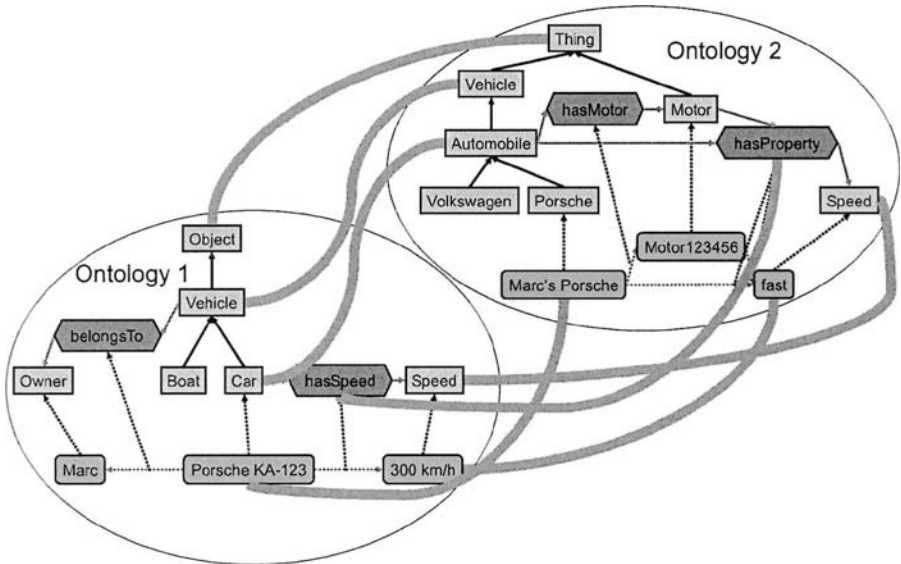


Fig. 2.2. Ontology Alignment Example

Table 2.1. Alignment Table

Ontology O_1	Ontology O_2	Confidence
object	thing	1.0
vehicle	vehicle	1.0
car	automobile	1.0
speed	speed	1.0
hasSpeed	hasProperty	1.0
Porsche KA-123	Marc's Porsche	1.0
300 km/h	fast	0.9

The following example illustrates alignments. The example consists of two simple ontologies that are to be aligned. The two ontologies O_1 and O_2 describing the domain of cars are given in Figure 2.2. The meanings of shapes, colors, and lines are the same as in the previous ontology example of Section 2.1.3. The first ontology has already been described in the ontology section; it is our running example. The second ontology covers the same domain but is modeled slightly differently. Beneath an overall thing concept, there exists a vehicle, which in turn has the subclasses automobile, Volkswagen, and Porsche. Further, there is a motor and speed. The automobile has a Motor which in turn has a property speed. A specific Porsche, Marc's Porsche, with the fast Motor123456 is also represented.

Reasonable alignments between the two ontologies is given in Table 2.1. Each line contains the two corresponding entities from ontology 1 and ontology 2. In Figure 2.2, alignments are represented by the shaded channels each linking two corresponding entities. Obviously, things and objects, the two vehicles, cars and automobiles, as well as the two speeds are the same. The relations of having a speed and property correspond to each other, as they both refer to speed. In addition, the two instances Porsche KA-123 and Marc's Porsche are the same, which are both fast. Whereas the alignments might seem obvious to the reader in this case, the common agreement on alignments is not easy in general and will be discussed more in the evaluation Section 5.4.

Returning to the formal definition of alignment, the set of result alignments is given as follows. To distinguish between the two ontologies, ontology 1 is given the namespace `o1` and ontology 2 the namespace `o2`.

$$align = \{o1:object \rightarrow o2:thing, o1:vehicle \rightarrow o2:vehicle, \dots\}$$

An excerpt of the alignments is shown in the RDF-based representation of Euzenat (2004) in Example 2.2. First the two ontologies and their URIs are defined. In the following cells the actual alignments are shown. The first one aligns `o1:object` with `o2:thing`. The confidence measure is very high with a value of 1.0, and the semantic relation between the two entities is equality (in contrast to subsumption or part of).

Example 2.2 (Example Alignment Representation in RDF(S)).

```

<rdf:RDF>
<Alignment>
  <xml>yes</xml>
  <level>0</level>
  <type>11</type>
  <onto1>ontology1.owl</onto1>
  <onto2>ontology2.owl</onto2>
  <uri1>http://aifb.uni-karlsruhe.de/ontology1.owl</uri1>
  <uri2>http://aifb.uni-karlsruhe.de/ontology2.owl</uri2>
  <map>
    <Cell>
      <entity1 rdf:resource='ontology1.owl#object' />
      <entity2 rdf:resource='ontology2.owl#thing' />
      <measure rdf:datatype='XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
    <Cell>
      <entity1 rdf:resource='ontology1.owl#vehicle' />
      <entity2 rdf:resource='ontology2.owl#vehicle' />
      <measure rdf:datatype='XMLSchema#float'>1.0</measure>
      <relation>=</relation>
    </Cell>
    ...
  </map>
</Alignment>
</rdf:RDF>

```

2.3 Related Terms

The focus of this work will be alignment of ontologies. Apart from alignment, there are many related terms, which will be defined and differentiated. The definitions are taken from Klein (2001), Ding et al. (2002), as well as de Bruijn et al. (2005). Unfortunately, the usage of the terms differs considerably.

Combining:

Two or more different ontologies are used for a task in which their mutual relation is relevant. The combining relation may be of any kind, not only identity. Therefore, no information on how the relation is established can be given at this point.

Integration:

For integration, one or more ontologies are reused for a new ontology. The original concepts are adopted unchanged, possibly, they are extended, but

their origin stays clear, e.g., through the namespace. The ontologies are merely integrated rather than completely merged. This approach is especially interesting, if given ontologies differ in their domain. Through integration, the new ontology can cover a bigger domain in the end. Ontology alignment may be seen as a prestep for detecting where the involved ontologies overlap and can be connect with each other. The most prominent integration approaches are union and intersection (Wiederhold, 1994), where either all entities of both ontologies are taken or only those which have correspondences in both ontologies.

Matching:

For matching, one tries to find two corresponding entities. These do not necessarily have to be the same. A correspondence can also be, e.g., in terms of a lock and the fitting key. A certain degree of similarity along some specific dimension is sufficient, e.g., the pattern of the lock/key. Whereas combining allows many different relations at the same time, matching implies one specific kind of relation. A typical scenario for matching is web service composition, where the output of one service has to match the corresponding input of the next service. Any schema matching or ontology matching algorithm may be used to implement the Match operator. Matching corresponds to our definition of general alignment, however, where a fixed relation between the aligned entities expresses the kind of match.

Mapping:

Ontology mapping is used for querying of different ontologies. An ontology mapping represents a function between ontologies. The original ontologies are not changed, but the additional mapping axioms describe how to express concepts, relations, or instances in terms of the second ontology. They are stored separately from the ontologies themselves. Often mappings can only be applied in one direction, e.g., the instances of a concept in ontology 1 may all be instances of a concept in ontology 2, but not vice versa. This is the case, if the mappings have only restricted expressiveness and the complete theoretical mapping relation cannot be found for the actual representation. A typical use case for mapping is a query in one ontology representation, which is then rewritten and handed on to another ontology. The answers are then mapped back again. Whereas alignment merely identifies the relation between ontologies, mappings focus on the representation and the execution of the relations for a certain task.

Mediation:

Ontology mediation is the upper-level process of reconciling differences between heterogeneous ontologies in order to achieve interoperation between data sources annotated with and applications using these ontologies. This

includes the discovery and specification of ontology alignments, as well as the actual usage of these alignments for certain tasks, such as mapping for query rewriting and instance transformation. Furthermore, the term ontology mediation also subsumes merging of ontologies.

Merging:

For merging, one new ontology is created from two or more ontologies. In this case, the new ontology will unify and replace the original ontologies. This often requires considerable adaptation and extension. Individual elements of the original ontologies are present within the new ontology, but cannot be traced back to their source. Alignment again is a prestep to detect the overlap of entities.

Transformation:

When transforming ontologies, their semantics is changed (possibly also changing the representation) to make them suitable for purposes other than the original one. This definition is so general that it is difficult to relate alignment to it.

Translation:

We here define translation as an operation restricted to data translation (Popa et al., 2002), which may also include the syntax, e.g., translating an ontology from RDF(S) to OWL. The representation format of an ontology is changed while the semantics is preserved. As we are talking about semantic alignments, translation is an underlying requirement when the formats differ, but we do not address translation itself.

This list consolidated common definitions, as they are used in this work.

2.4 Ontology Similarity

Similarity plays a central role for ontology alignment in this work. Here, the meaning of similarity needs to be explained. Thereafter, similarity for ontologies is given a general framework (Ehrig et al., 2005a) followed by specific assigned similarities.

2.4.1 Ontology Similarity Definition

Ontology similarity, as used in this work, refers to the comparison of whole ontologies or subelements thereof. This comparison returns a numerical value indicating whether the two elements have a high or low degree of similarity. This can be formally laid down through the following definition.

Definition 2.8 (Ontology Similarity). *A similarity function*

$$\text{sim} : \mathfrak{P}(E) \times \mathfrak{P}(E) \times O \times O \rightarrow [0, 1]$$

is a function that maps a pair of entity sets (expressed through the power set $\mathfrak{P}(E)$ of entities) and their corresponding ontologies O to a real number expressing the similarity between two sets such that

- $\forall e, f \in \mathfrak{P}(E), O_1, O_2 \in O, \text{sim}(e, f, O_1, O_2) \geq 0$ (positiveness)
- $\forall e, f, g \in \mathfrak{P}(E), O_1, O_2 \in O, \text{sim}(e, e, O_1, O_2) \geq \text{sim}(f, g)$ (maximality)
- $\forall e, f \in \mathfrak{P}(E), O_1, O_2 \in O, \text{sim}(e, f, O_1, O_2) = \text{sim}(f, e, O_2, O_1)$ (symmetry)
- $\forall e, f \in \mathfrak{P}(E), O_1, O_2 \in O, \text{sim}(e, f, O_1, O_2) = 1 \Leftrightarrow e = f$: Two entity sets are identical.
- $\forall e, f \in \mathfrak{P}(E), O_1, O_2 \in O, 0 < \text{sim}(e, f, O_1, O_2) < 1$: Two entity sets are similar/different to a certain degree.
- $\forall e, f \in \mathfrak{P}(E), O_1, O_2 \in O, \text{sim}(e, f, O_1, O_2) = 0 \Leftrightarrow e \neq f$: Two entity sets are different and have no common characteristics.

Different similarity measures $\text{sim}_k(e, f, O_1, O_2)$ are indexed through a label k . As before, we leave out O_1, O_2 when they are evident from the context and write $\text{sim}_k(e, f)$ instead. e and f may be sets of concepts, relation, or instances. This includes subtrees or even whole ontologies. A set may also consist of only one entity, thus in the extreme case reducing the similarity to a similarity between two individual elements.

As our goal alignment relation is equality, and equality is a symmetric relation, we adhere to a symmetric similarity, although we are aware that it is controversially discussed (Mitra and Wiederhold, 2001). Humans tend not to follow the symmetry rule when they decide on similarity between two objects (Bernstein et al., 2005a). Normally, the similarity between object 1 and object 2 is rated in the context of object 1, whereas for similarity between object 2 and object 1 the context is object 2.

2.4.2 Similarity Layers

Since an ontology represents a conceptualization of a domain, comparing two ontology entities goes far beyond the representation of these entities only (syntax level). Rather, it should take into account their relation to the real world entities they are referencing, i.e., their meaning, as well as their purpose in the real world, i.e., their usage. In order to achieve such a comprehensive comparison, we use a semiotic view (theory of signs) on ontologies and define our framework for similarity in three layers, as shown in Figure 2.3: Data-, Ontology-, and Context Layer (Ehrig et al., 2005a). The arrangement of these layers already indicates that they build upon each other. We enhance these by an additional orthogonal dimension representing specific domain knowledge. Initial blueprints for such a division in layers are found in the semiotics

(theory of signs) for example in Maedche and Staab (2002), where they are called symbolic, semantic, and pragmatic layer, respectively. Other classifications focused on matching, but clearly related to the notion of similarity, are presented in Rahm and Bernstein (2001) and Shvaiko and Euzenat (2005).

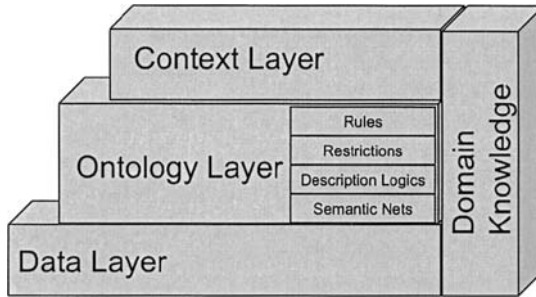


Fig. 2.3. Similarity Layers

Data Layer:

On this first layer, we compare entities by only considering data values of simple or complex datatypes, such as integers and strings. To compare data values, we may use generic similarity functions such as the edit distance for strings. For integers, we determine a relative distance between them. Complex datatypes made up of simple datatypes also require more complex measures, but which are effectively compiled of simple measures.

Ontology Layer:

For the second layer, the ontology layer, we consider semantic relations between the entities. In fact, one may split this layer again along the semantic complexity, which is derived from the layer cake of Berners-Lee (2000). On the lowest level, we just treat the ontology as a graph with concepts and relations. These Semantic Nets were introduced by Quillian (1967). This level is enhanced by description logics like semantics (Baader et al., 2003), e.g., a taxonomy is created over concepts, in which a concept inherits all the relations of its superconcepts. For example, if certain edges are interpreted as a subsumption hierarchy, it is possible to determine the taxonomic similarity based on the number of is-a edges separating two concepts. Besides intensional features, we also rely on the extensional dimension, i.e., assess concepts to be the same, if their instances are similar. For restrictions, e.g., in the ontology language OWL, we again use different heuristics. Higher levels of the ontology layer cake also become interesting for similarity considerations. Especially if similar rules between entities exist, these entities will be regarded as similar.

For this type of similarity, one has to process higher order relationships. Unfortunately, there has not been sufficient research and practical support for the rule layer in the Semantic Web in general, even less for similarity considerations. However, first work has been done in this direction through Fürst and Trichet (2005). The similarity functions of the ontology layer fall back on similarity functions of the data layer.

Context Layer:

On this layer, we consider how the entities of the ontologies are used in an external context. This implies that we use information external to the ontologies. We consider contexts as local models that encode a party's subjective view of a domain. Although there are many contexts in which an ontology can be considered (for example the context in which an ontology is developed, or in which it has been changed), from the point of view of determining the similarity, the most important one is the application context, e.g., how a specific entity of an ontology has been used in the context of a given application. An example for this is the Amazon portal¹¹ in which, given information about which people buy which books, one may decide if two books are similar or not. Therefore, the similarity between two ontology entities is easily determined by comparing their usage in an ontology-based application. A naïve explanation is such that similar entities have similar patterns of usage. The main problem remains how to define these usage patterns (Stojanovic, 2005) in order to discover the similarity in the most efficient way. To generalize the description of such patterns, we reuse the similarity principle in the terms of usage: Similar entities are used in similar context. We use both directions of the implication in discovering similarity: If two entities are used in the same (related) context, then these entities are similar and vice versa; if in two contexts the same (related) entities are used then these contexts are similar.

Domain Knowledge:

Special shared ontology domains have their own additional vocabulary, e.g., the bibliographic domain often relies on the Dublin Core (Caplan, 1995) constructs `dc:author` or `dc:title`. This domain specific vocabulary is treated in a special way, thus returning own similarities, i.e., an author similarity or title similarity instead of a general attribute similarity. The right part of Figure 2.3 shows the domain-specific aspects. As this domain-specific knowledge may be situated at any level of ontological complexity, it is presented as a vertical box across all of them.

2.4.3 Specific Similarity Measures

After having presented the general similarity layers, we now show specific measures, which fit the framework. The list of measures is by no means

¹¹ <http://www.amazon.com/>

complete, but shall give an overview of important individual similarities. They will be reused later on in this work for identifying ontology alignments. An extensive work, not necessarily focused on ontology similarity only, has been provided by Bernstein et al. (2005b).

Data Layer

In the ontology definition section we have defined datatypes to be concepts, and their concrete values being instances $V \subset I$. These value instances allow special operators for comparison based on the individual letters of a string or the numerical value of an integer. The lexical signs (typically strings) for concepts, relations, and instances ($G = G_C \cup G_R \cup G_I$), are also compared based on data layer similarities. In this section we will not distinguish between V and G , thus $v \in (V \cup G)$.

Equality:

For some datatypes and scenarios, it is appropriate to require equality of data values for entities to be similar. An example where this is necessary would be data values that are used as identifiers. v_1 and v_2 are respective data values.

$$sim_{equality}(v_1, v_2) := \begin{cases} 1, & \text{if } v_1 = v_2, \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

Syntactic Similarity:

Levenshtein (1965) introduced a widely-used measure to compare two strings, the so-called edit distance. The idea behind this measure is to take two strings and determine how many atomic actions are required to transform one string into the other one. Atomic actions are addition, deletion, and replacement of characters, but also shifting their position. The number of operations are expressed in *ed*. For our purposes, we rely on the syntactic similarity of Maedche and Staab (2002), which is inverse to the edit distance measure.

$$sim_{syntactic}(v_1, v_2) := \max(0, \frac{\min(|v_1|, |v_2|) - ed(v_1, v_2)}{\min(|v_1|, |v_2|)}) \quad (2.2)$$

Names are a prominent example, which require syntactic similarity. Small variations such as different cases or typing errors still lead to a high similarity. Many other possibilities to compare strings exist, e.g., Stoilos et al. (2005). Often it will also be necessary to use bag-of-words approaches when the label does not only contain one word but a whole description. Chaves (2003) for instance uses stemming (Porter, 1980) and the sequence of words instead of characters to determine the similarity.

Distance-Based Similarity for Numeric Values:

For numeric datatypes with a limited range (e.g., subsets of integer or double) it is advisable to dispose of a similarity measure that assigns the similarity between two numerical values based on their arithmetical difference (Bergmann, 2002). A generic example of such a distance-based similarity measure is

$$sim_{diff}(v_1, v_2) := 1 - \left(\frac{|v_1 - v_2|}{maxdiff} \right)^\gamma \quad (2.3)$$

where $v_1, v_2 \in [\min_{v \in V}(V), \max_{v \in V}(V)]$ and $maxdiff = \max_{v \in V}(V) - \min_{v \in V}(V)$ for a numeric datatype V . The parameter $\gamma \in \mathbb{R}^+$ may be used to lessen or amplify the influence of increasing differences on the similarity assessment.

Objects

Whereas on the data layer we simply compare data values with each other, we now define how to actually compare objects, without respecting their semantic type yet. In our similarity framework objects only exist on the upper layers making object similarities part of the ontology layer.

Object Equality:

Object equality of ontology objects is based on existing logical assertions. This assertion may be explicitly included in the ontology definition, e.g., `owl:sameAs`, or have been identified at an earlier point in time either manually or by an automatic approach.

$$sim_{object}(e, f) := \begin{cases} 1 & align(e) = f, \\ 0 & \text{otherwise} \end{cases} \quad (2.4)$$

Dice Coefficient:

Often it is necessary to compare not only two entities but also two sets of entities E and F . Two sets of entities may be compared based on the overlap of the sets' individuals ($e \in E, f \in F$) (Castano et al., 1998):

$$sim_{dice}(E, F) := \frac{2 \cdot |E \cap F|}{|E| + |F|} \quad (2.5)$$

Unfortunately, the dice coefficient only returns results if the individuals are marked with a unique identifier where equal individuals have the same identifier, even across several ontologies. Only then, it is possible to exactly determine whether an individual is in only one or both of the sets.

Jacquard Coefficient:

This coefficient is related to the previous set similarity. It calculates the fraction of overlapping elements compared to the number of all existing elements in the two sets.

$$sim_{jacquard}(E, F) := \frac{|E \cap F|}{|E \cup F|} \quad (2.6)$$

Single Linkage:

The maximum similarity between two sets is a basic operator for comparisons. Linkage measures are typically used for measuring distances of sets for data mining, e.g., for clustering. Here we apply them for similarities instead. In contrast to the coefficient similarities, single linkage also deals with similarities of individual entities. In specific, it determines the maximum. For this, we rely on other measures that already did the calculation of similarity values $[0, 1]$ between single entities.

$$sim_{single}(E, F) = \max_{(e,f) | e \in E, f \in F} (sim(e, f)) \quad (2.7)$$

Average Linkage:

Many other related similarities, e.g., based on average similarities between the sets' individuals are also possible.

$$sim_{complete}(E, F) = \frac{\sum_{(e,f) | e \in E, f \in F} sim(e, f)}{|E| \cdot |F|} \quad (2.8)$$

Multi Similarity:

This measure compares sets by representing them through an average element each. As the individual entities have various different features, it is difficult to create a feature vector representing the whole sets. Therefore, we use a technique known as a prestep to multidimensional scaling. We describe each entity through a vector representing the similarity to any other entity contained in the two sets. For both sets, a representative vector can be created by calculating an average vector over all individuals. Please note, that this averaging means that we cannot distinguish any longer, if one entity pair of the sets was very similar or many entity pairs were only slightly similar. Finally, we determine the cosine between the two set vectors through the scalar product as the similarity value.

$$sim_{multi}(E, F) = \frac{\sum_{e \in E} \mathbf{e}}{|\sum_{e \in E} \mathbf{e}|} \cdot \frac{\sum_{f \in F} \mathbf{f}}{|\sum_{f \in F} \mathbf{f}|} \quad (2.9)$$

with $\mathbf{e} = (\text{sim}(e, e_1), \text{sim}(e, e_2), \dots, \text{sim}(e, f_1), \text{sim}(e, f_2), \dots)$, \mathbf{f} analogously.

This measure has several advantages. In contrast to any similarity based on maximum considerations, multi similarity is monotonic and thus more suited for optimization. A differing number of elements in the sets does not distort the similarity, e.g., when only one element needs to be aligned with many elements, still the whole range of $[0,1]$ for similarity may occur. Vice versa, the equality of only two elements in the sets does not lead to an overall high similarity.

For more set similarities we refer to Valtchev (1999); Brinkhoff (2004); Bernstein et al. (2005a).

Ontology Layer

We continue with specific similarity measures derived from the ontology structure. As for the other layers, these measures are examples; many others are also possible.

Label Similarity:

One basic feature of entities in ontologies is their label. Labels are human identifiers (names) for entities, normally shared by a community of people speaking a common language. We rely on string similarity to compare the labels, in our case syntactic similarity.

$$sim_{label}(e, f) := sim_{syntactic}(label(e), label(f)) \quad (2.10)$$

Dictionaries (WordNet (Fellbaum, 1998)) may further be used for comparisons even across languages. Two strings representing synonyms return 1, otherwise 0. Please note that homonyms would erroneously also return high similarities.

Taxonomic Similarity for Concepts:

One possible generic measure to determine the semantic similarity of concepts C within one ontology, in one concept hierarchy \leq_C , has been presented in Rada et al. (1989):

$$sim_{taxonomic}(c_1, c_2) := \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, & \text{if } c_1 \neq c_2, \\ 1, & \text{otherwise} \end{cases} \quad (2.11)$$

$\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h in the concept hierarchy, respectively. The shortest path length is a metric for measuring the conceptual distance of c_1 and c_2 . The intuition behind using the depth of the direct common subsumer in the calculation is that concepts at upper layers of the concept hierarchy are more general and are semantically less similar than concepts at lower levels. This measure is easily used analogously for relation R comparisons through \leq_R .

Extensional Concept Similarity:

Besides describing a concept $c \in C$ through its features (intension), using the extension is also feasible. Two concepts are similar, if they have similar instances I . We use a set similarity for comparing the sets of instances, e.g., multi similarity.

$$sim_{extension}(c_1, c_2) := sim_{multi}(\iota_C(c_1), \iota_C(c_2)) \quad (2.12)$$

Domain and Range Similarity:

We define a similarity measure for relations $r \in R$ based on their domain and range definitions given by σ_R :

$$\begin{aligned} sim_{domRan}(r_1, r_2) := & 0.5 \cdot (sim_{object}(ran(r_1), ran(r_2)) \\ & + sim_{object}(dom(r_1), dom(r_2))) \end{aligned} \quad (2.13)$$

Concept Similarity of Instances:

Instances I have a certain similarity if they are assigned to the same parent class through ι_C .

$$sim_{parent}(i_1, i_2) = sim_{object}(c_1, c_2) \text{ with } i_1 \in \iota_C(c_1), i_2 \in \iota_C(c_2) \quad (2.14)$$

As there are many more ontological structures, which are used to determine the similarity of entities, this list can easily be extended. In fact, the general approach for ontology alignment in this book will refer to some additional ones later on.

Context Layer

The similarity of two objects on the contextual level is defined as:

$$sim_{use}(e, f) := sim_{diff}(Usage(e, con), Usage(f, con)), \quad (2.15)$$

where $e, f \in E$ are entities from the ontology model (e.g., two concepts or two instances) and $Usage(e, con)$ corresponds to the frequency of the usage of e in the context con . As we already mentioned, the context could be a portal application in which the given entities are used. Note that this formula can be easily extended in order to compare the usage of some selected characteristics of two entities, instead of the entities themselves. It enables several levels of abstractions in which the similarity can be discovered. For example, two books are similar if their authors (that are different persons) have many coauthored publications.

Especially the two areas of context similarity and domain knowledge will not be extended here as a reasonable description cannot be given in general, but requires a fixed application scenario. Section 8.2 will present such similarities for one specific case, the Bibster application.

2.4.4 Similarity in Related Work

Similarity measures for ontological structures have been widely researched, e.g., in cognitive science, databases, software engineering and artificial intelligence. Rodríguez and Egenhofer (2000) give a general overview of similarity. In Bisson (1995) we find measures that deal both with the local structure of the objects and the relational structure that exist between the objects thus approaching some of the issues of our ontology layer. In Bisson (1992) the attention is restricted to the comparison of concepts. Furthermore, Bisson does not distinguish relations into taxonomic relations and other ones, thus ignoring the semantics of inheritance. Weinstein and Birmingham (1999) compute description compatibility in order to answer queries that are formulated with a conceptual structure that is different from the one of the information system. Their measures depend largely on a shared ontology that mediates between locally extended ontologies. This is related to our domain-specific measures, which make use of a shared ontology. Their algorithm also seems less suited to evaluate similarities of sets of lexical entries, taxonomies, and other relations.

Bernstein et al. (2005a) presented interesting results on how humans interpret similarity. They assembled a catalogue of ontology-based similarity measures, which have experimentally been compared with a similarity gold standard obtained by surveying 50 human subjects. Results show that similarity predictions among humans and among algorithms varied substantially, but can be grouped into cohesive clusters. It will be interesting to apply these findings for alignment in future.

2.4.5 Heuristic Definition

The main operator for comparisons is similarity. Moreover, the listed concrete similarities are sufficient for most cases of ontology alignment as considered here. However, it is also possible to use other more general operators, which are here summarized through the term heuristics. In the end, they are a generalization of similarity as required for general or complex ontology alignment (Section 10.2).

Definition 2.9 (Heuristic). *A heuristic for a relation ρ is a function*

$$heur_\rho : \mathfrak{P}(E) \times \mathfrak{P}(E) \times O \times O \rightarrow [0, 1]$$

- $\forall e, f \in \mathfrak{P}(E), heur(e, f, O_1, O_2) \geq 0$ (*positiveness*)
- $heur_\rho(e, f, O_1, O_2) = 1$: *The relation ρ holds between the two objects.*
- $0 < heur_\rho(e, f, O_1, O_2) < 1$: *The relation ρ holds to a certain degree.*
- $heur_\rho(e, f, O_1, O_2) = 0$: *The relation ρ does not hold.*

Heuristics measure to which degree a relation such as inclusion (on sets, strings, or numbers), overlap, or also dissimilarity is met. A complete list

of heuristics and their concrete implementation for ontologies is beyond the focus of this work, as they are only required for few methods of identity ontology alignment. The corresponding implementations will be defined in the respective sections when they are required.

An example for a heuristic on numerical values is the *smaller than* relation. In this case, we adhere to a fuzzy notion of the relation *smaller than*. Comparing two prices might result in:

$$smallerThan(price(e), price(f)) = smallerThan(9, 10) = 0.8 \quad (2.16)$$

In contrast to existing work, the similarity layers present a concise structure for different similarity measures and help users to find the ones suiting their need best. We will make use of it for our ontology alignment.



<http://www.springer.com/978-0-387-32805-8>

Ontology Alignment

Bridging the Semantic Gap

Ehrig, M.

2007, XVIII, 248 p., Hardcover

ISBN: 978-0-387-32805-8