Ontology-Based Image Annotation and Retrieval

Avril Styrman

Master of Science Thesis
Instructors: Eero Hyvönen & Hannu Erkiö
University of Helsinki
Dept. of Computer Science
Helsinki 19th November 2004

Contents

1 Introduction **1**
- 1.1 Annotation and Retrieval . . . 1
- 1.2 Semantic Interpretation of Image . . . 2
- 1.3 Semantic Web Ideology . . . 4

2 Image Annotation and Retrieval Techniques **5**
- 2.1 Text-Based Paradigm . . . 5
  - 2.1.1 Text-Based Annotation . . . 6
  - 2.1.2 Text-Based Retrieval . . . 7
- 2.2 Field-Based Paradigm . . . 9
  - 2.2.1 Field-Based Annotation . . . 9
  - 2.2.2 Field-Based Retrieval . . . 10
- 2.3 Structure-Based Paradigm . . . 11
  - 2.3.1 Structure-Based Annotation . . . 11
  - 2.3.2 Structure-Based Retrieval . . . 12

3 Ontologies **13**
- 3.1 Theory of Ontological Categories . . . 13
  - 3.1.1 Historical Overview . . . 13
  - 3.1.2 Category Tree . . . 14
  - 3.1.3 Contrasts in Categorization . . . 17
  - 3.1.4 The Principle of Triads . . . 19
  - 3.1.5 Annotation and Retrieval . . . 20
  - 3.1.6 Conclusions . . . 21
- 3.2 Formal Ontologies . . . 22
  - 3.2.1 Ontology Types and Frameworks . . . 23
  - 3.2.2 Semantic Web Standards . . . 24

4 Case Study on Ontologization, Annotation, and Retrieval **29**
- 4.1 Ontologization . . . 29
  - 4.1.1 Requirement Analysis . . . 29
  - 4.1.2 Languages and Tools . . . 30
  - 4.1.3 Prototype Process Model . . . 31
  - 4.1.4 Acquiring the Domain Knowledge . . . 32
  - 4.1.5 Selecting Top Level Classes . . . 33
  - 4.1.6 Working up the Ontology . . . 34
  - 4.1.7 Testing the Ontology . . . 36
  - 4.1.8 Analysis of the Created Ontology . . . 37
- 4.2 Ontology-Based Image Annotation . . . 46
  - 4.2.1 Constraining the Annotation . . . 46
  - 4.2.2 Annotation with the Created Ontology . . . 49
- 4.3 Ontology-Based Image Retrieval System . . . 54
  - 4.3.1 Using Inference in Recommending Photos . . . 55
  - 4.3.2 Structure-Based Search . . . 56

5 Conclusions **62**

Acknowledgments **67**

References **68**

Appendix A: Content-Based Image Retrieval CBIR **78**
Naive CBIR 78
Computer Vison 78
Machine Vision 79
Towards Integrating Formal Ontologies with CBIR 80

Appendix B: General Concepts of Image Analysis **82**
Icon − Index 82
Factual Content − Truth Content 84
Sign − Signification 84
Denotation − Connotation 84
Punctum − Studium 84

Appendix C: Examples of Ontologies **85**
Plato's Categories 85
Tree of Porphyry 86
Wilkins' Categories 88
Brentano's Tree 90
CYC 91
Wordnet 92
Open Directory Project 92
Sowa's Diamond 94

1 Introduction

The aim of this thesis is to describe how ontologies can be used to create better image annotation and retrieval systems. In a nutshell, ontologies are used to overcome the problems that evolve from traditional text-based information retrieval when it is applied to images.

Text-based information retrieval is lexically motivated rather than conceptually motivated, which leads to irrelevant search results in information retrieval. Lexically motivated means that text-based retrieval operates on the word-level, and not on the level of the meaning of words. The very idea of ontologies is that they are conceptually motivated, i.e., can be used to express the intended meaning of things, and not just words as textual strings.

The common techniques that are developed for document retrieval in general may be applied to metadata-based image retrieval without modification, and also the techniques and methods represented in this thesis in the scope of image retrieval can be applied to document retrieval in general. Hence, the majority of the methods represented in this thesis can be applied to all information retrieval.

In this thesis, a domain model is built using a formal ontology language, and the model is used as the information source and basis of the information retrieval structure of a photograph exhibition system. The domain model and the exhibition system are examined in detail. The domain model that is built is called a domain ontology, and ontologies are explained starting from the earliest philosophical notions until to the latest scientific publications concerning image retrieval.

1.1 Annotation and Retrieval

Annotation stands for the process of describing images, and retrieval stands for the process of finding images. The two major approaches to image retrieval are *content-based* image retrieval that analyzes the actual image data, and *metadata-based* approach that retrieves images based on human-annotated metadata. Also relevance feedback has been used in image retrieval complementing text-based systems. In this thesis the retrieval is done by using the annotated metadata, and not the content-based analysis or relevance feedback. The research problem is: "How should the metadata be created and what kind of system could interpret the metadata to make it easy to find images for an average user?"

The metadata that describes images can be roughly divided in two parts. One part concerns the concepts that give information about the creator of the image, tools used in the process of creating the image, art style of the image and the artist, price, and other explicit properties of the image. The other part describes what is actually *in* the image, the implicit properties that can be understood by percepting the image itself. These two parts cannot be cleanly separated, and both have to be taken into account when analyzing an image.

1.2 Semantic Interpretation of Image

There are no commonly agreed vocabularies or methods to analyze images among museums' curators, art critics, semiotics, aesthetics, philosophers, artists, and scientists. An overview of the terms used in image analysis is given in appendix B, and the central concepts around perception of image are briefly discussed in the following.

Figure 1 depicts the entities that are present whenever an image is percepted visually. The perceptor (spectator) confronts (tuché) an image (spectrum) created by an artist

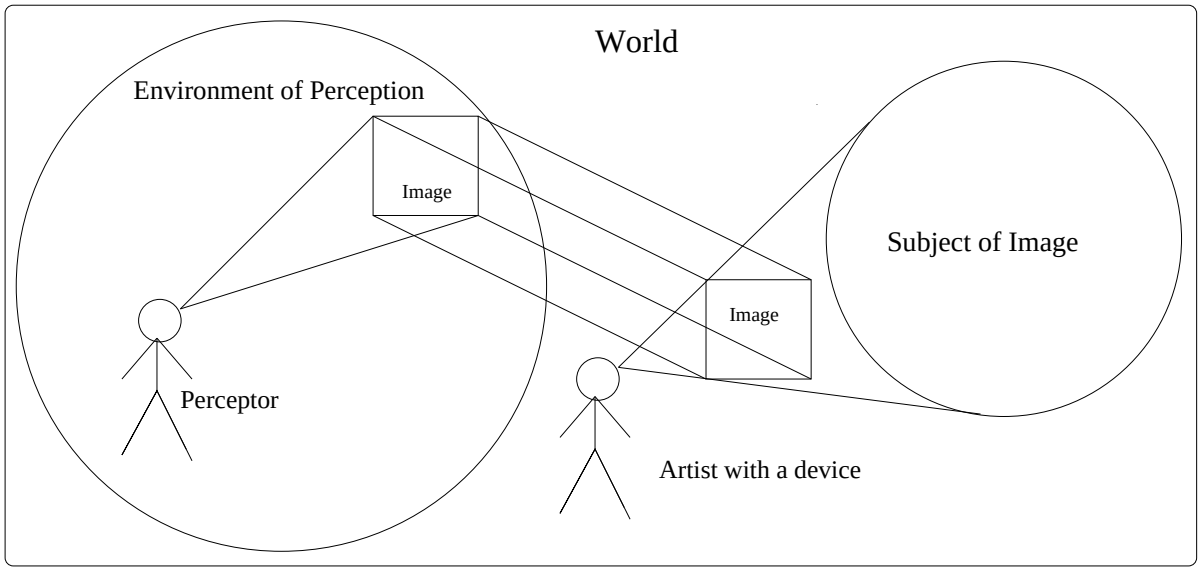


Figure 1: The artist constrains a view of the world with a device, according to Lachan's idea of photography [136].

(operator) [12]. The environment of perception is the place where the image is seen and includes all the factors that have affect on the perceptor's interpretation[1]

The perceptor sees the *physical elements* of the image. With printed images the physical elements constitute of the ink and the paper where the ink is printed on, or only of the ink if the paper is transparent. In a negative the physical elements are the silver-nitrates that form the image and in digital images the finite set of pixels P with specified RGB- and other values. A single physical element might cover the total area of the image but in the usual case many clearly distinguishable subsets of P can be picked out. After the physical elements have been percepted the perceptor subconsciously connects them to a set of *abstract elements*, learned concepts in the perceptor's inner model of the world. Physical elements are in relation with abstract elements, which is called here *resemblance*.

[1]One fundamental part of the environment other than the image itself is the possible reference text. Reference text is different from possible text *in* the image [50], which would definitely be part of the image, but relationships with the image and an unattached text vary between a short title and an exhaustive analysis [85].

Figure 2 is examined as an example of interpreting an image. The combined features of the face resemble `smile`. The hands are geometrically upper than the head and the body, which resembles `hands upwards`. The person is wearing sports-clothes with a number and commercial tags on them. These together with the audience resemble `competition`. `Smile`, `hands upwards`, and `competition` all together resemble strongly a `triumph`.

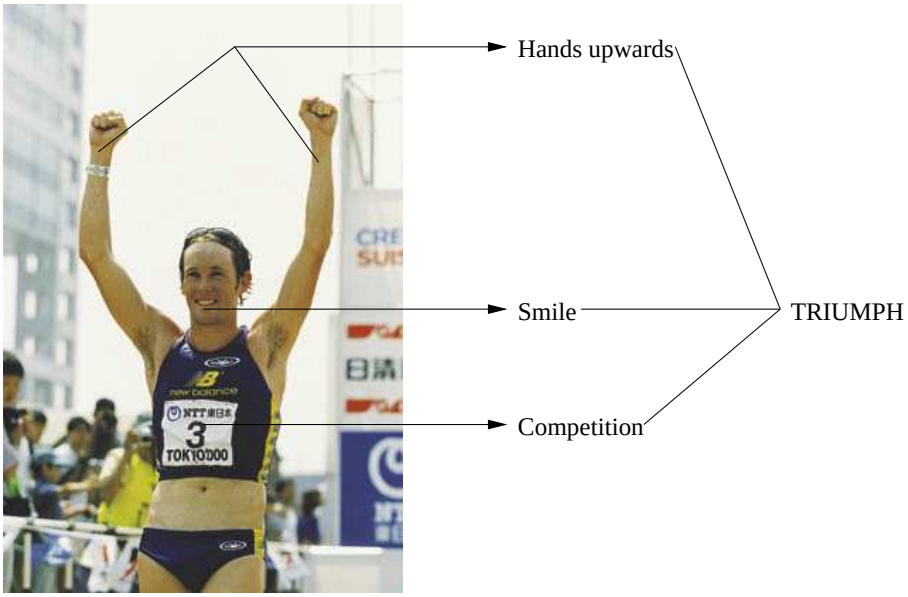


Figure 2: A photograph representing triumph.

It could be said that `triumph` is a higher-order resemblance than the others because it is derived from `smile`, `hands upwards`, and `competition`. Depending on the abstraction level, none of the abstract elements marked on figure 2 directly resemble the physical elements. It is easy to analyze images on this level of abstraction but doing the same thing automatically with computers is very hard (appendix A). This is why only the abstract elements specified by human annotators are used as the basis of image retrieval and annotation in this thesis.

There are so many possible resemblances of the physical elements that any image, even an empty one, can have myriads interpretations. This indicates that no image can be annotated absolutely perfectly, i.e., it is essential that the annotator has a certain intension in the annotation, and knows approximately how the annotations will be used. If the intended usage of the annotations is unclear, the annotator should annotate also the direct resemblances in addition to the clearest indirect resemblances. When all the direct resemblances have been annotated, semantic ontologies can be used to reason about the indirect resemblances without further manual annotations. Ontologies can also be used to specify that a concept such as `triumph` consists of many possible permutations of resemblances of different physical and abstract elements such as `smile` and `hands upwards`, and so an image that has only one annotation like `triumph`, probably contains some general elements that `triumph` consists of. Therefore, ontologies provide a promising aid in semantic image retrieval.

1.3 Semantic Web Ideology

The Semantic Web techniques can be used used to reason about the relations of the abstract elements of images. The idea of the Semantic Web [17, 135] is to use the arising XML- and RDF-based standards (sect. 3.2.2) to serve as common norms of all information representation and description on the Internet. These languages and frameworks are recommendations of the *World Wide Web Consortium W3C* [158] that is a very influential international consortium in the Semantic Web field. Ontologies are used as the new means of creating and using the metadata in annotation and retrieval. In addition to searching just character strings within a natural language text, intended meanings of words and contexts surrounding plain-text representations can be created and used with ontology-based languages and tools.

Ontology frameworks have gained popularity since the 90's with an increasing pace. Ontology research is a hot topic in today's computer science and the development of the Semantic Web is tied with ontologies. Ontologies provide an easy and feasible way of capturing a shared understanding of terms that can be used by humans and programs to aid in information exchange. Computer science gives us the technique to express ontologies efficiently in a useful way, and Internet is the ideal testing ground for ontology-based applications.

The contents of this thesis are introduced in the following:

Section 2 gives an overview to digital photograph archival techniques. Examples of annotating and retrieving images with different paradigms are given.

Section 3 discusses the common theoretical aspects of ontologies as well as modern formal ontology frameworks and languages with the scope on standard Semantic Web ontology languages.

Section 4 gives a case-study about using ontologies in practice. The section is divided into 1) creation of an ontology, and using the created ontology in 2) image annotation and in 3) image retrieval.

Section 5 discusses what has been accomplished and evaluates the usefulness of ontology-based approach in image annotation and retrieval, and in building a domain model.

Appendices further examine some aspects of the actual thesis. Appendix A examines content-based image retrieval and how ontologies can be integrated with the content-based image retrieval. Appendix B examines the terms that are used in image analysis. Appendix C examines some well known philosophical and formal ontologies.

2 Image Annotation and Retrieval Techniques

This section gives an overview to the techniques that are used in digital image annotation and retrieval, but may the data be in any form, there are fundamental similarities in the process of annotating and retrieving it.

Different organizations do not share a generally accepted standard description model to be used with photograph collections [89]. Even though most of the organizations use standards approved by ISO [81] or ANSI [3], there is a wide variety of guidelines ranging from specific photography guidelines such as FOTIOS [48], library standards like AACR2 [1] and ISBN [80], to archival standards like ISAD(G) [79] and general thesauri such as YSA [161]. Institutions have also developed their own guidelines. One explanation of this patchwork of different guidelines is that institutions aim to include photographic materials into their general description and cataloging system. An exclusive standard for photographic materials might conflict with specific institutional guidelines, and in this case the same search facility is used for all materials.

Even thought the used systems are very heterogeneous, they use features of the three basic image retrieval paradigms, which can be seen as generic types of the techniques that are actually used with applications. In the following these three generic paradigms [155] are discussed and examples of annotation and retrieval are given using images about the *promotion happening*, that is also the subject domain of the case example in section 4. *Text-based*, *field-based*, and *structure-based* paradigms are presented in order of simplicity and can be seen as extensions of one another. The *Content-based* paradigm differs greatly from the others because it is not metadata-based, and so it is explained in appendix A.

2.1 Text-Based Paradigm

Text-based (also called keyword-based) methods are so general that they have to be used up to some degree with all the paradigms. Everything can be described with natural language but it is hard to solve the intended meaning of a textual description automatically with computers. This is why different semantically oriented techniques have been created to support text-based information retrieval.

2.1.1 Text-Based Annotation

Annotating images with the text-based paradigm is very simple. The annotator has to write a textual description of a photo using natural language. After the description has been created it is linked to an image. The image text of figure 3 corresponds to a text-based annotation, and contains information about both the explicit and implicit properties (sect. 1.1) of the image. To facilitate the retrieval the annotators have to take

Figure 3: Promotion of faculty of Philosophy 1886. General garland binder Hedvig Estlander posing with her garland. Photographing Studio: Daniel Nyblin

in account the possible use of *thesauri* that constrain and guide the use of vocabulary. A thesaurus is a collection of natural language words that specifies the vocabulary of some specific domain. Anything can be described in myriads of ways if all the words of English or another commonly used language are in use. With a vastly narrower set of words the annotations are surely more homogeneous and the creation of queries is a much simpler task because the used vocabulary is known.

Thesauri may contain relationships of words such as synonyms but specifying more complex relationships such as homonyms, hyponyms, meronyms, and antonyms require a richer level of formality than thesauri usually have. With thesauri it is also easier to map existing descriptions between different languages because only those words that are included in the thesaurus have to be used. There are a number of domain-specific thesauri available such as the *Art and Architecture Thesaurus* [118] that has vocabulary to describe art, architecture, decorative arts, material culture, and archival materials. The coverage of the AAT ranges from Antiquity to the present, and the scope is global. The Library of Congress Subject Headings [98] is a thesaurus that aims in classification of uniform and unique headings, provision of direct access to specific subjects, stability, and consistency. An image-centered Iconclass [76] is a collection of ready-made definitions of objects, persons, events, situations and abstract ideas that can be the subject of an image. There are thesauri (or lists) also for very common words (e.g., `in`, `a`, `and`, `the`, `for`) to exclude these from a search.

Assuming that the thesaurus is descriptive enough for the domain, the only disadvantage is that the annotators and retrievers have to check out the valid words with an explicit thesaurus browser, i.e., they cannot just write what comes first in mind.

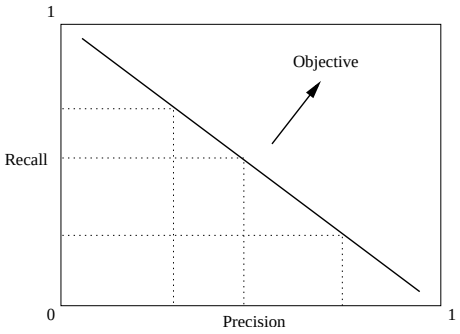
2.1.2 Text-Based Retrieval

In the retrieval sense, photos that are annotated with plain text behave similarly to plain text documents because both contain text, which can be exploited by conventional text-based retrieval techniques.

The generic text-based information retrieval is carried on so that first a user types a query that consists of 1 to n keywords into a query field of a search interface. The search engine compares the keywords with a set of documents gathered from a database and gives them priority values. For example, if the keyword is `book`, document A contains two instances of `book` and document B contains only one instance, then A gets a higher priority. The documents are presented to the user, highest priority first.

When the size and amount of the documents grows, the classical problems of text-based information retrieval start raising. Irrelevant documents are retrieved and the user has to use time filtering the information again, usually by browsing through the search results. The fitness of generic text-based retrieval is depicted in figure 4 [86]. When the recall gets higher the precision gets lower, and when precision gets higher the recall gets lower.

	Relevant	**Irrelevant**
Found		
a+b	a	b
Discarded		
c+d	c	d
Sum		
a+b+c+d	a+c	b+d



$$recall = \frac{a}{a+c} \qquad precision = \frac{a}{a+b}$$

$$1 \geq recall \geq 0 \qquad 1 \geq precision \geq 0$$

Figure 4: Symbols `a`, `b`, `c`, and `d` represent the members of the documents that are subject of the search. The graph represents a usual relation between recall and precision. The fitness of recall and precision can be calculated with the equations.

Some techniques that facilitate the text-based search are discussed next. The main principle of the *vector-model* [133, 22] is to index documents that consist of text into a matrix including statistical information such as the amount of appearances of different words in a document and the location of the words in the document. This is radically faster than to search directly from the documents, even with optimized *character parsing techniques* [93, 19]. The vector-model also provides the search algorithms a structured basis to start the reasoning with.

Stemming uses morphological relations of words to find a part[2] of a word that is common to all or most of the forms of the word. For example, if the query term `running` is used with the stemming-option on, the engine would not just search for `running` but also `run`, `runnable`, `runner`, etc. A user can create queries such as `run*` that have the same effect

[2]In most of the cases the beginning of words.

as the stemming. The unknown * can be also situated in the beginning, in the middle, or in any place within a word or a phrase, possibly many times within one word or phrase.

With *boolean search* one can define the wanted relations of the search terms with logical connectives *and* $\wedge$, *or* $\vee$, *not* $\neg$. The user can also set search constraints to find documents where a certain word is near, before, or after another.

These kinds of techniques have been created to overcome the bottlenecks that evolve from keyword-based search but they are only 'pain-killers', not a 'cure' to the problem. Because these techniques are not alone sufficient, other more semantically oriented techniques have been taken in use in addition to the thesauri. There are *probability models* used for reasoning about documents' usefulness. These models analyze phrase constructs and take the structure of a document in account. Many documents, especially Web pages, have *hyperlinks* to other documents. Search engines (e.g., Google [55], CiteSeer [24]) can reason about documents' usefulness not only by indexing them but also by researching other documents to find the most referenced ones.

As an example of text-based image retrieval, the retriever's goal is to find images where can be seen a general garland binder and garland in the same image[3]. The retriever types the keywords: `General garland binder` $\wedge$ `Garland`. The search engine compares the keywords `general`, `garland`, and `binder` to all image annotations in the system. A set of annotations match with the query because all of them contain the keywords. Some of the result are depicted in figure 5.

Figure 5: A subset of images that matched the query.

A large percentage of the images in figure 5 have no garland, which indicates a relatively low precision. The basic problem is that text-based search does not even try to understand the meaning of words and sentences, which is quite natural: understanding of a textual string would be an unsolvable problem for a machine because a correct interpretation of a short phrase such as a query or an annotation could be impossible also for humans without any knowledge of the retriever's goal or the specific context under which the query should be handled.

Another common approach to making text-based information retrieval more robust involves keyword expansion using a thesaurus or other lexical resource. However, keyword expansion using thesauri is limited in its usefulness because keywords expanded by their synonyms can still only retrieve documents directly related to the original keyword. Furthermore, a naive synonym expansion may actually contribute more noise to the query and negate what little benefits keyword expansion may give [103]: if keywords cannot

[3]Protégé-2000 [124] was used in this example as explained in p. 59.

have their meaning solved, then all synonyms of a particular keyword may be used in the expansion, and this has the potential to retrieve many irrelevant documents.

Attempting to overcome the limited usefulness of keyword expansion by synonyms, various researchers have tried to use slightly more sophisticated resources for query expansion. These include dictionary-like resources such as lexical semantic relations [153], keyword co-occurrence statistics [119, 102], as well as resources generated dynamically through relevance feedback, like global document analysis [160], and collaborative concept-based expansion [91] that also requires the use of relevance feedback [105].

Although some of these approaches are promising, they share some of the same problems as naive synonym expansion. Dictionary-like resources such as WordNet (appendix C) and co-occurrence frequencies, although more sophisticated that just synonyms, still operate mostly on the word-level and suggest expansions that are lexically motivated rather than conceptually motivated. Relevance feedback, though somewhat more successful than dictionary approaches, requires additional iterations of users and cannot be considered a fully automated retrieval.

2.2 Field-Based Paradigm

The field-based approach (also called attribute-based and feature-based) describes and retrieves an item by one or more field-value pairs. This way the field-based paradigm can be seen as an extension to the text-based paradigm where only one field is used in annotation and retrieval. Typically a metadata schema (ontology) is defined that describes a set of fields and some indication is given about the type of values that can be assigned to a particular field. The most widely used schema for describing on-line documents in general is the *Dublin Core* metadata template [28]. The fields of DC version 1.1 are `title`, `creator`, `subject`, `description`, `publisher`, `contributor`, `date`, `type`, `format`, `identifier`, `source`, `language`, `relation`, `coverage`, and `rights`. Qualified versions of DC have been created for specialized domains such as for describing art objects in museums [156].

2.2.1 Field-Based Annotation

The usual case is that the annotator has a number of fields where the required values can be set. Some fields take text as value and some take values like integer, boolean, date, etc. Some values might be more or less predefined. There might be a menu with a number of color options associated to a field, and properties like *length* could be constrained into the metric system. Many systems require giving proper values to certain fields before accepting an annotation. Then again all fields do not have to be given values because these might not be essential or there possibly does not exist values for all the fields with all the annotated items.

As with the text-based paradigm, having an agreed thesaurus simplifies both the retrieval and annotation processes. Many of the field-based initiatives recommend the use of closed thesauruses such as the AAT (sect. 2.1.1) but do not associate particular parts of a thesaurus with a field. As a consequence the only support that a human annotator has is some sort of thesaurus browser or a reference book. To improve the support for annotation a mapping is required from the fields to particular parts of the thesaurus so

that the annotator is only presented with terms that are relevant for a particular field. This would again be very close to more semantically oriented structure-based paradigm.

The example case is to annotate the same photo (fig. 3, p. 6) that was annotated with the text-based paradigm. The annotator has the image, image text, and a set of empty fields to start with. Some fields in the below table describe the implicit properties or the image (topic, color), some describe the explicit properties (photographer, copyrights), and some fields describe both (reference words, other information), just as explained in section 1.1.

`photographer`	`date of photography`	`topic`	`reference words`
`negative size`	`color`	`annotator`	`copyrights`
`other information`	`image type`	`image size`	`condition`
`archival`			

Some fields might be hard to understand without any additional info. The `Topic` field corresponds to the whole annotation of the text-based paradigm in figure 3, with the exception of the photographing studio that could be inserted in field `photographer` or `other information`. When all the required fields have been properly filled the annotation is done.

2.2.2 Field-Based Retrieval

The retrievers usually do not have to give values to all the fields of the search interface to find the wanted images and can be totally unaware of the system beforehand, when in contrast the annotators are usually trained for the job. When the annotator sets a certain value to a field like integer 1, the retrievers can possibly query a certain range of values such as integers between [0 10], all dates between years 1500-2000, and character strings just as with the text-based paradigm. Again, the use of an agreed vocabulary helps to find the wanted images, and in contrast, using incorrect vocabulary might make the search very hard. Fields with pre-defined value specifications are very helpful when the vocabulary used in the annotation process in unknown. It would be very time-consuming and unnecessary to always give values to all the possible fields if there are many of these. One simple way of executing a field-based search is to first give values to only a few fields and start the search. If the precision is too low the retriever can set more constraints by giving values for a few more fields or giving more accurate values for some fields.

However, an average user who is not interested in the explicit properties of the image uses mostly the topic field. In this case the retrieval is identical to text-based retrieval. The only difference is the possibility to set values for a number of other fields, which can have other kinds of value types. The planning of the fields is again very close to ontology-based planning.

2.3 Structure-Based Paradigm

Structure-based paradigm [155] can be seen as an extension to the field-based paradigm. Where the field-based approach essentially uses a flat structure of attribute-value pairs, the structure-based approach allows more complex descriptions involving relations. For example, a description of a piece of furniture can include a description of it's components, e.g., a drawer of a chest. The components are again objects that can be described using a number of attributes such as material, size, and shape. Components can even have components themselves, e.g., drawers can have handles, and theoretically the specification of the subcomponents can go up to the depth where a component can not have any more specific subcomponents.

Ontology-based design is the only way to construct structure-based systems, and in general structure-based systems can be called ontology-based systems. The methods explained here are on very general level and ontologies (sect. 3) have to be understood to thoroughly understand the structure-based paradigm.

2.3.1 Structure-Based Annotation

There is a vague line between what is called a field-based or a structure-based annotation schema, but one fundamental difference to field-based paradigm is the method of selecting the values for the fields. With field-based systems the values are most often natural language nouns typed with the keyboard, but structure-based systems allow selecting the values from within *category trees* like the one in figure 6. The category trees can be built using formal ontology languages.

Categories can be equated with the traditional folders (or directories) of a personal computer and the individual images that the categories describe can be equated with the files that are stored in the folders of a PC. The traditional folder structures are only more constrained than the categories: the folders can have many subfolders but only one parent folder - categories can have many subcategories and many parent categories; the same file can belong to one folder only - an individual can belong to many categories.

Let the subject of the example annotation be again the image on figure 3 in p. 6 and let the fields be the same as with the field-based paradigm in sect. 2.2.1, with the exception that the values of field `topic` are selected in a structured way. The annotator has the image and the image text to work with. Based on this information the annotator has to select values for the `topic` field from within the category tree on figure 6. The tree contains three categories: *Happenings*, *Persons*, and *Objects*. In every category there is a list of individuals in an alphabetical order: *Happenings* contains many individual happenings, `Persons` contains many individual persons, and `Objects` contains many individual objects.

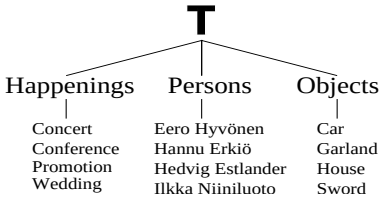


Figure 6: A category tree from which values are selected for the `topic` field.

The annotator selects *Promotion* from category *Happenings*, *Hedvig Estlander* from category *Persons*, and *Garland* from category *Objects*. In case the wanted individuals would not be on the lists the annotator could add them there. If the annotator would not know that the person was Hedvig Estlander, she/he could have annotated only the category *Persons* to tell that there is some person in the image.

The example was very simple, but relational descriptions can vary widely between different categories of things and the annotation can be carried on in myriads of ways. It is clear that annotation has to be handled in a way that different annotators can use the same annotation interface at different times successfully not doing the same work again, seeing what has already been annotated, and possibly supplementing the previously made annotations. With relatively small or simple schemas annotations can be done very easily and swiftly but with complex schemas the process can get harder.

2.3.2 Structure-Based Retrieval

Values were selected from a category tree in the annotation. In the retrieval the same tree is used in finding the images that are linked to certain categories or individuals. Structure-based retrieval can be done in very many different ways just as the annotation. Implementation of the retrieval system naturally depends on the way the annotations are made but the most essential principles in the retrieval are set theoretic *intersection* $\cap$, *union* $\cup$, and *difference* $\setminus$ [4].

The retrieval process can be carried on by first choosing a promising category from within a set of top level categories and following a promising path from a category to another until the wanted kind of category or individual is found. When $\mathbf{x}$ (category or individual) is selected, the system retrieves all images that were linked to $\mathbf{x}$ in the annotation stage. By selecting *Hedvig Estlander* from the category tree the system retrieves all images that were linked to *Hedvig Estlander* in the annotation.

The intersection $\mathbf{x}_1 \cap \mathbf{x}_2$ retrieves those, and only those images that are linked to both $\mathbf{x}_1$ and $\mathbf{x}_2$. The union $\mathbf{x}_1 \cup \mathbf{x}_2$ retrieves the images that are linked to $\mathbf{x}_1$ or to $\mathbf{x}_2$. The difference $\mathbf{x}_1 \setminus \mathbf{x}_2$ retrieves the images that are linked to $\mathbf{x}_1$ but not to $\mathbf{x}_2$. The search constraints can also be combined. The expression $(\mathbf{x}_1 \cap \mathbf{x}_2) \setminus \mathbf{x}_3$ retrieves images that are linked to both $\mathbf{x}_1$ and $\mathbf{x}_2$, but not to $\mathbf{x}_3$.

Structure-based methods have been used for example with the *HiBrowse* system [123] in the 90's, and more recently with *Flamenco* [35], *Promotor* (sect. 4.3), and *MuseumFinland* [74] for example. In addition to these academic projects, structure-based methods have been taken in use in an increasing pace also with every-day applications because of their outstanding performance. For example *INOA* search [78], *Open directory project* (appendix D), and *Soneraplaza topic search* [139] all allow constraining the search with categories. The 'structure' in the structure-based paradigm can be equated with ontological categories, and ontologies are examined in the next section.

[4] These principles are explained in more detail in sect. 3.1.5

3 Ontologies

This section briefly introduces ontologies to the reader with the scope limited to categorization, i.e., in relating categories with each other in the same way as with formal Semantic Web ontologies.

Theoretical basis of ontologies [142, 138, 34] are reviewed first in section 3.1 in order to clarify the questions about meaning of categories and the basis of describing things with categories. The classes of Semantic Web ontologies, as well as the categories of philosophical ontologies (appendix C) are in a predecessor-successor hierarchy, which facilitates using intersection, union, and difference to explain the logic in frame-based categorization, and in structure-based annotation and retrieval in simple means.

Section 3.2 examines modern formal ontologies with the scope on the standard Semantic Web ontology languages. Frame-based modeling [96] is currently the most common design principle of Semantic Web ontologies and categorization is the backbone of frame-based modeling. Ontology languages such as RDFS and OWL support frame-based modeling, which is also used in the case example in section 4 in creating a domain ontology.

3.1 Theory of Ontological Categories

The history of ontological categories is briefly discussed in section 3.1.1. Any rationally constructed category tree follows a certain logic, that is deterministically explained in section 3.1.2 by equating the categories with ZF-sets[5]. The theory is applied in practice in section 3.1.3, where the *Physical-Abstract* and *Continuant-Occurrent* divisions are explained with visual category tree representations in order to show the relativeness of any categorization. In section 3.1.4, *triads* (Firstness, Secondness, Thirdness) are used to explain the principle of how relations of categories are expressed with RDF, that is the basis of formal Semantic Web ontology languages (sect. 3.2.2). Section 3.1.5 shows how category trees can be used to annotate and retrieve information.

3.1.1 Historical Overview

Word *ontology* comes from the Greek words *ontos* (study of being) and *logos* (word). On some occasions ontology is treated as a synonym for metaphysics, domain, or context [137]. The word was used in Christian theology in scope of examining God's metaphysical appearance throughout the middle ages, and was taken in use also in philosophy in the 17th century [64, 104, 32, 47]. Philosophical ontology is the science of what is, of the kinds of structures, categories of objects, properties, events, processes, and relations in every area of reality. The taxonomies which result from philosophical ontology have been intended to be definitive in the sense that they could serve as answers to questions such as: "What categories of entities are needed for a complete description and explanation of all the going-ons in the universe?" In this sense a perfect ontology should cover all types of entities, including also all the *relations* by which the entities are associated with each other. *A priori* knowledge, being something we know or can reason about without having to percept it again can be equated with ontology; ontology represents our a priori knowledge. Even though the term was used in theology, philosophers like

[5]Zermelo-Fraenkel set theory is a generally accepted formalization of set theory.

Plato, Heraclitus, and Aristotle examined the metaphysical structure of the world in the ancient Greece long before the birth or Christianity.

3.1.2 Category Tree

The problem of *Universals* and *Particulars* can be seen as an important stage of early ontology research. A universal can be defined as an abstract object or term that ranges over particular things; *roundness* ranges over all things that are round like stars and planets. Universals can be equated with ontological categories and particulars can be equated with the things that the categories describe: the universe, and categories lower in the hierarchy.

The division to universals and particulars is problematic in nature, and the line between an universal and a particular depends on a viewpoint. From a human point of view, the individual particulars are those things that a human would approximately consider as a unique assemblance of physical and abstract things that is so unique that there cannot be two absolutely identical of the same kind, or if there are, these too can be enumerated and called individuals. The particulars are often concrete, like individual humans, stars, and planets, but also all the categories can be called individuals. Every category and every ontology is unique: if there are two ontologies that are not identical, then they are individual, and if two ontologies are identical, then they are the same individual ontology. And every category is unique, because there is no other category in it's specific place in some ontology, of which all are unique. All that can be said is that categories that are higher in a hierarchy of an ontology are more universal than those lower in the hierarchy, because those categories that are lower in the hierarchy describe a more condensed view of reality than the ones above them. The line between universals and particulars is drawn where it is the most practical to draw from the scope of an average person.

The categories in figure 7 were specified already by Plato even though he used the concept *divided line* [122] instead of a category tree representation. The terms used with the category tree are introduced in the following with Plato's categories.

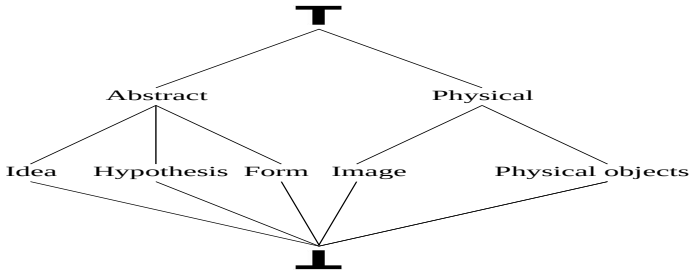


Figure 7: Plato's categories represent the universals.

- **Universal type $\top$ and primitive $\bot$**
  $\top$ is the root category that is always on the top of the tree. $\top$ contains all differentiae and describes everything. The primitive or absurd type $\bot$ is the only leaf category of the tree. $\bot$ contains only what is common to all categories, and describes only what is common to all the things that all the categories in the tree describe.

- **Subcategory**
  Having two categories connected with a line, the category that is geometrically lower than the other is subcategory of the upper category. Subcategory of an arbitrary category X describes a more condensed view of the reality than X, relative to what X describes. *Abstract* and *Physical* are subcategories of $\top$. *Idea*, *Hypothesis* and *Form* are subcategories of *Abstract*. *Image* and *Physical objects* are subcategories of *Physical*. $\bot$ is subcategory of *Idea*, *Hypothesis*, *Form*, *Image*, and *Physical objects*.

- **Supercategory**
  Supercategory is an inversion of subcategory. Having two categories connected with a line, the category that is geometrically higher than the other is supercategory of the lower category. Supercategory of an arbitrary category X describes a wider view of the reality than X, relative to what X describes. $\top$ is supercategory of *Abstract* and *Physical*. *Abstract* is supercategory of *Idea*, *Hypothesis* and *Form*. *Physical* is supercategory of *Image* and *Physical objects*. *Idea*, *Hypothesis*, *Form*, *Image*, and *Physical objects* are supercategories of $\bot$.

- **Successor**
  Successors of an arbitrary category X are all those categories that are confronted by following the line downwards from a category to another, starting from X. All successors of X describe a more condensed view of the reality than X, relative to what X describes. All categories are successors of $\top$ except $\top$ itself. $\bot$ is a successor of every category except $\bot$ itself, and no category is successor of $\bot$.

- **Predecessor**
  Predecessor is an inversion of successor. Predecessors of an arbitrary category X are all those categories that are confronted by following the line upwards from a category to another, starting from X. All predecessors of X describe a wider view of the reality than X, relative to what X describes. $\top$ is predecessor of every category except $\top$ itself, and no category is predecessor of $\top$. Every category is predecessor of $\bot$, except $\bot$ itself.

Category tree is a lattice[6], but since drawing of the greatest node $\top$ and the smallest node $\bot$ is not enforced it can be called a tree as well. The relations of categories could be described with predicate logic or with any other suitable formalism, but the simplest category tree requires only the $supercategoryOf$ relation and can be taken as a boolean algebra [54] model $\mathcal{M} = \{\mathcal{E}, >\}$, where $\mathcal{E}$ stands for the categories and $>$ stands for the $supercategoryOf$ connective between the categories[7].

A category can have a totally different meaning when placed as a subcategory of different categories. Especially the meaning of iconic signs (Appendix B) like *Circle* changes along the supercategory: as a subcategory of *Form*, *Circle* would stand for an abstract circular form, and as a subcategory of *Physical objects* it would stand for a physical object that has a circular form. In addition to the category itself, relations to other categories determine the meaning of a category. In case of model $\mathcal{M}$, *predecessors and successors of an arbitrary category X determine the meaning of X*. A category X describes its

[6]Lattice is simply a tree where there is always a path from every node (or category) to the greatest node $\top$, and to the smallest node $\bot$.

[7]Using model $\mathcal{M}$, *Abstract* $>$ *Form* means that *Abstract* is supercategory of *Form*. The notation is useful whenever describing category hierarchies within text.

successors, but the successors also describe category X; predecessors of X describe X, but X also determines the meaning of it's predecessors.

In terms of ZF set theory, the categories can be equated with *sets*, where subcategories are *proper subsets* of their supercategories[8]. The set theoretic relations of categories are visualized with figure 8. The categories of the second and the third tree from the left are in correspondence with the first tree from the left: in the second tree $A = \{1, 2\}$ and $B = \{2, 3\}$; in the third tree $A = \{1, 2, 3\}$ and $B = \{2, 3, 4, 5\}$.

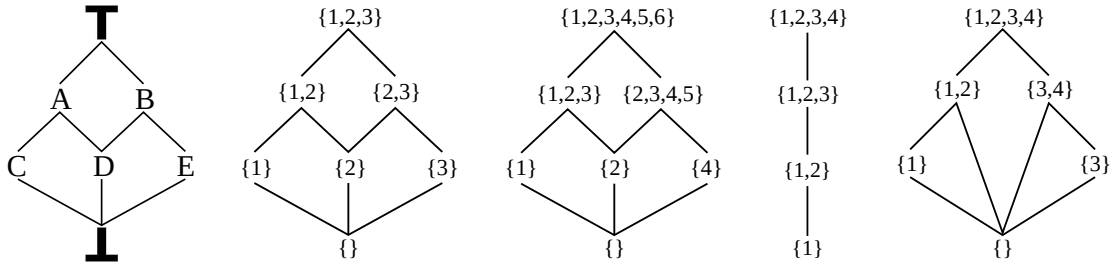


Figure 8: Examples of category trees where the categories are depicted as ZF sets.

When an arbitrary category X has two or more subcategories, the union of these is a subset of X. A and B are proper subsets of $\top$: $A \subset \top$, $B \subset \top$. The union of A and B is a subset of $\top$: $A \cup B \subseteq \top$. It is important to understand that $A \cup B = \top$ does not have to hold because $A \cup B$ can be also a proper subset of $\top$: in the second tree from the left $A \cup B = \top$, $C \cup D = A$, $D \cup E = B$, and $C \cup D \cup E = \top$, but in the third tree from the left $A \cup B \subset \top$, $C \cup D \subset A$, $D \cup E \subset B$, and $C \cup D \cup E \subset \top$, i.e., all categories contain more than the unions of their subcategories contain.

When X has two or more supercategories, X is a subset of the intersection of these. D is a subset of the intersection of A and B: $D \subseteq A \cap B$, i.e., D can contain all that is common to A and B or only a proper part of it. In the second tree from the left $D = A \cap B$, but in the third tree from the left $D \subset A \cap B$.

In the second tree from the right the category that corresponds to $\bot$ is not empty. A *strict definition* is applied in the rightmost tree, where all subcategories of X are disjoint, i.e., an intersection of two or more successors of X that are not in predecessor-successor relation is empty: when X has two subcategories x_1 and x_2, $x_1 \subseteq X \backslash x_2$, and $x_2 \subseteq X \backslash x_1$, which implicates that $x_1 \cap x_2 = \{\}$.

The set theoretic definition enables understanding the boundaries of ontologies. A perfect philosophical ontology should have the means to describe everything that exists, but the limit of the descriptive power of any ontology is that an ontology cannot be thoroughly described with itself. This limitation can be derived directly from *Russell's Paradox* [110], formulated by Bertrand Russell (1903). All categories can be equated with sets, and all categories of an ontology are proper subsets of $\top$, except $\top$ itself. When an ontology describes itself, $\top$ should be a proper subset of $\top$, which is impossible. The conclusion is that an ontology can only describe things external to the present state of itself.

[8]When inner sets are not allowed, all members of the sets are $\in$-minimal and cannot be further divided. This way, also all successors of an arbitrary category X have to be proper subsets of X. When the cardinality of $\top$ is c, the maximum depth of the tree is c, and the maximum amount of different categories is 2^c.

3.1.3 Contrasts in Categorization

The reality is relative and so is the ontology that describes it: categories classify things relative to other categories. All categories are in contrast and describe different types of properties of things. Two dichotomies of top-level categories are examined in the following to clarify the use of the set theoretic definition in practice.

Dichotomy of Physical and Abstract

Successors of *Abstract* describe abstract properties of things, and successors of *Physical* describe physical properties of things in figure 9. For example, the abstract form of an iron ball is *Round* and it is made of physical *Metal*. A frisbee is also *Round*, but it is made of *Plastic* instead of *Metal*.

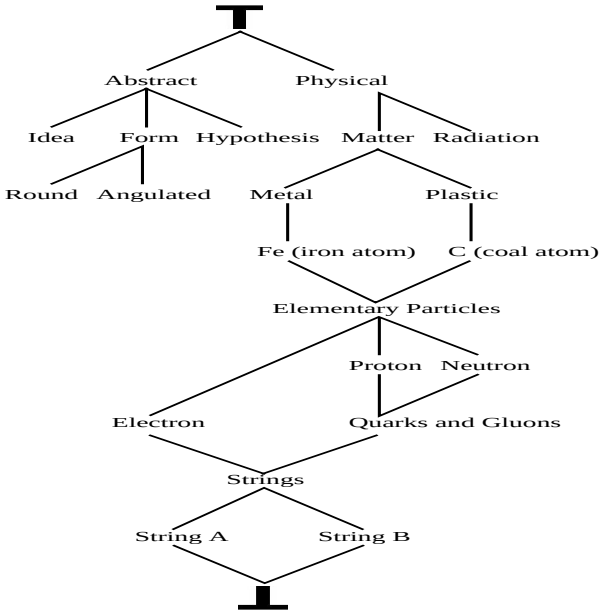


Figure 9: Plato's categories of figure 7 in p. 14 are modified and the categorization is not intended to be definitive. Most of the successors of *Abstract* are depicted as leaves of the tree because of representational reasons, but $\bot$ is still a successor of all of these.

The strict definition is applied in the division to *Physical* and *Abstract*: when a category is successor of *Abstract*, it can not be successor of *Physical*[9]. Things that are generally considered concrete, touchable, visible, or measurable such as *Matter* and *Radiation* are described with successors of *Physical*. *Abstract* things like information structures in living beings' minds or in the memory of a computer are opposite to *Physical* things. These kinds of things are untouchable, invisible, that possibly cannot be explained such as *Form* and *Idea*. Many of the other categorizations are not disjoint, such as the division of *Matter* to *Metal* and *Plastic*.

When an arbitrary category X has one or more subcategories, then X is something where all of these belong to: *Physical* and *Abstract* belong to $\top$, *Matter* and *Radiation* belong to *Physical*, and *Metal* and *Plastic* belong to *Matter*. X can contain more than it's

[9]Unless the categories and the individuals are equated.

subcategories, but X can also constitute of it's subcategories only. $\top$ contains more than just *Physical* and *Abstract*, *Matter* contains more than *Metal* and *Plastic*, but *Elementary Particles* constitutes exactly of *Electron*, *Proton*, and *Neutron*.

When X has one or more supercategories, then X is something that belongs to all of these: *Elementary particles* belongs to both *Fe* (iron atom) and *C* (coal atom) because they both constitute of the same *Elementary Particles*. *Proton* and *Neutron* then again constitute of *Quarks and Gluons*. String-theory discusses how *Electron*s and *Quarks and Gluons* constitute of *Strings*.

The intersection of *String A* and *String B* is marked with the unknown $\bot$ because it is not known what is common to two different strings. $\bot$ is what is common to all of the categories in the tree, that is intuitively nothing.

Dichotomy of Continuant and Occurrent

In addition to *Physical* and *Abstract*, also other categories can be applied as high in the hierarchy. The division to *Continuant* and *Occurrent* classifies the continuity of things relative to each other in figure 10.

All matter tends to change within a period of time, including the planets, galaxies, and the whole known universe. A continuant has a stable set of properties that describe its various appearances at different times to be recognized as the same thing. The most continuant things are placed as subcategories of *Continuant* and the least continuant things as subcategories of *Occurrent*. Relative to being a *Continuant*, *Planet* is subcate-

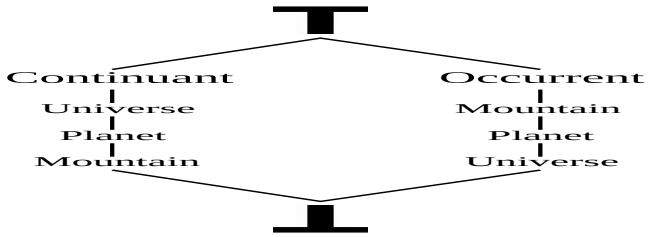


Figure 10: Division to *Continuant* and *Occurrent*.

gory of *Universe* and *Mountain* is subcategory of *Planet*. The bottom-most successors of *Continuant* (before $\bot$) in the tree describe things that remain recognizable as the same thing for the shortest period of time.

Relative to being an *Occurrent*, *Planet* is subcategory of *Mountain* and *Universe* is subcategory of *Planet*. The bottom-most successors of *Occurrent* would then describe the entities that remain recognizable as the same thing for the longest period of time.

When two things are as continuant or occurrent they can be placed as subcategories of the same category. The intersection of successors of *Continuant* and *Occurrent* is marked with $\bot$. In this case 'absurd type' describes $\bot$ quite well because it is hard to imagine what is common to successors of both *Continuant* and *Occurrent*.

3.1.4 The Principle of Triads

The idea of triads (Firstness *I*, Secondness *II*, and Thirdness *III*) was born when modern philosophers[10] were creating triadic category patterns as means to create new categories from the previously created ones. *I*, *II*, and *III* have been also included in a philosophical ontology *as* categories[11], but here the principle is used in explaining the relations of categories.

According to Peirce (1891) "First is the conception of being or existing independent of anything else. Second is the conception of being relative to, the conception of reaction with something else. Third is the conception of mediation, whereby the first and the second are brought into relation."

The triadic pattern is widely applicable in explaining relations of things, and is used in fields such as philosophy, art, semiotics, and cognitive science to describe thinking, formation of perception, and interpreting art. The principle can be used also in explaining the meaning of a category when its supercategory is known. *I* can be taken as a holistic punctum (appendix B): a category that comes first in mind and appears to be independent. *II* is the domain, context, or supercategory that surrounds *I*, and *III* is the meaning of *I* in the context of *II*. For example, when *I* is *Form* and *II* is *Abstract*, then *III* is the meaning of *Form* in context of *Abstract*, that is an abstract form. When *I* is *Form* and *II* is *Application*, *III* would be totally different: an application form.

Next the triadic pattern is used in categorization with a naive example. Categories *Earth*, *Mars*, and *Pluto* are subcategories of $\top$ in the tree on the left of figure 11. *Earth*

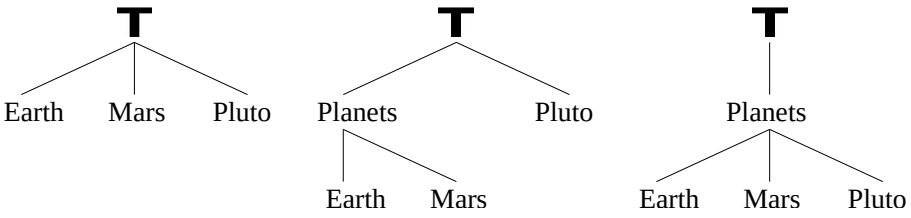


Figure 11: Example of categorization with the principle of triads.

and *Mars* are selected as *I* and *II*. They both are *Planets* (*III*), and the result is seen on the center. Also other category pairs, *Earth-Pluto* and *Mars-Pluto* could have been selected as well as *Earth-Mars*. The final step is to select pair *Pluto-Earth* or *Pluto-Mars* as *I* and *II*, which both are *Planets* (*III*), and so, also *Pluto* can be set as a subcategory of *Planets*.

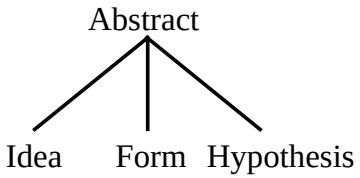
Placing the new categories as high in the hierarchy as possible is in theory sufficient to create any categorization because any categorization can be done one triad at a time. In addition of creating new categories out of the existing ones the principle of triads can be used in reducing the number of categories. Two similar categories (*I* and *II*) can be combined into one (*III*), when the new category collects successors of *I* and *II*, and therefore *I* and *II* can be discarded from the tree. When only one category X is discarded, X's supercategories collect the successors of X. If category *Planets* was discarded from the tree on the right of figure 11, the result would be the tree on the left of figure 11.

[10]Kant, Peirce, Husserl, Whitehead, Heidegger, and others.

[11]Sowa's Diamond in appendix C.

Has-test [142] can be used when it is difficult to decide whether a category suits as a subcategory of another category by inserting 'X has Y'. If the pattern sounds normal, then Y can be placed as a subcategory of X. 'Has' can be replaced with many words and nouns such as 'has greater expressive power than' or 'suits for describing a wider area of reality than'. When X and Y change places, other kinds of versions of the test can be used like 'Y is part of X', or 'Y belongs to X'. For example, '*Metal* fits for describing wider area of reality than *Iron*' and '*Iron* is part of *Metal*' sound natural, and so category *Iron* can be placed as a subcategory of *Metal* rather than the other way round. In some cases it is intentional to use the test in both directions, like 'Y is X' and 'X is Y'. '*Iron* is *Metal*' sounds reasonable, but '*Metal* is *Iron*' does not.

The principle of triads is applied also in the very basis of the Semantic Web. In RDF (sect. 3.2.2) information is expressed in form of triads, or triples: `Subject-Object-Predicate`, or `Subject-Property-Value`. On the left side of figure 12 is a category tree and on the right side is a table with three triples that are in correspondence with the tree. Naturally, also other kinds of properties can be expressed with triples, but the *supercategoryOf* relation '¿' is used as an example. The triple $(Abstract, >, Form)$



Subject	Property	Value
Abstract	>	Idea
Abstract	>	Form
Abstract	>	Hypothesis

Figure 12: A category tree and the corresponding triples.

tells that *Abstract* is supercategory of $Form$. Any member of the triple can be queried when two out of three members are known. The known members can be taken as I and II, when III is the result of the query. The query $(Abstract, >, III)$ returns $Idea$, $Form$, and $Hypothesis$, i.e., all `values` that *Abstract* has for `property` $>$. The queries $(Abstract, III, Idea)$, $(Abstract, III, Form)$, and $(Abstract, III, Hypothesis)$ all return $>$, i.e., the `property` of *Abstract* that has `values` $Idea$, $Form$, and $Hypothesis$. The queries $(III, >, Idea)$, $(III, >, Form)$, and $(III, >, Hypothesis)$ all return *Abstract*, i.e., the `subject` that has `values` $Idea$, $Form$, and $Hypothesis$ for `property` $>$. Two or more triples can be unified, which allows more complex queries.

3.1.5 Annotation and Retrieval

The fundamental principles of structure-based annotation and retrieval [134] can be explained with category tree using the set theoretical notation. The task is to annotate six items, or instances, denoted by members of the annotation set $\{1, 2, 3, 4, 5, 6\}$ with the given category tree on the left side of figure 13. The annotation is done by linking subsets of the annotation set to the categories. Let the triple, or query $(Y, annotationOf, III)$ return the annotations of category Y. The result of the query (III) is abbreviated as Y_a in the following. On the left side of figure 13 is the category tree without annotations and for example $A_a = \{\}$ and $H_a = \{\}$[12].

[12]Note that the sets depict only the values of $annotationOf$ properties in this example and not the categories that remain the same.

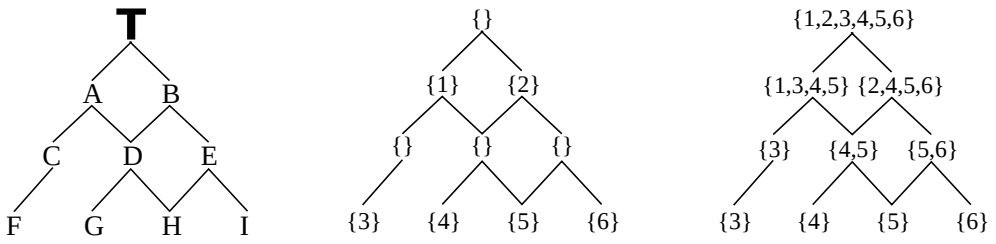


Figure 13: Sets represent annotations of categories $A, B, C, D, E, F, G, H, I$.

The annotation is executed and the result can be seen in the tree on the center. For example, $A_a = \{1\}$ and $F_a = \{3\}$, but $\top_a = \{\}$ and $D_a = \{\}$. The retrieval is only a matter of how the annotations are used, i.e., rules for retrieval have to be defined. Let the rule be: "annotations of subcategories of an arbitrary category X are unified with the annotations of X." Hereby, predecessors inherit the annotations of all their successors. This rule is based on the fact that predecessors describe what is common to all of their successors: all things that are described with successors of *Roundness* are also round. In the tree on the right the annotations have been inherited all the way up to the top. For example, union of G_a and H_a constitutes D_a: $G_a \cup H_a = \{4\} \cup \{5\} = \{4, 5\} = D_a$. The rules concerning the sets that depict the annotations of the rightmost tree are the same as with the set theoretical definition of the category tree with one exception: annotation of subcategory of X does not have to be a proper subset of X_a, but can also be the same as X_a. For example, $F \subset C$ but $F_a = \{3\} \subseteq \{3\} = C_a$. Examples of retrieval are given in the following.

- **Intersection** of two of more annotations retrieves those, and only those images that are members of all participants of the intersection. When intersection of A_a and B_a is selected, items 4 and 5 are retrieved: $A_a \cap B_a = \{1, 3, 4, 5\} \cap \{2, 4, 5, 6\} = \{4, 5\}$.
- **Union** of two or more annotations retrieves all those images that are members of any of the participants of the union. When union of A_a and B_a is selected, items 1,2,3,4,5, and 6 are retrieved: $A_a \cup B_a = \{1, 3, 4, 5\} \cup \{2, 4, 5, 6\} = \{1, 2, 3, 4, 5, 6\}$.
- **Difference**. Selecting difference of one annotation retrieves all images except those that are members of the selected annotation. Difference of two annotations retrieves the images that are members of one but not of the other annotation. When difference of A_a and B_a is selected, items 1 and 3 are retrieved: $A_a \backslash B_a = \{1, 3, 4, 5\} \backslash \{2, 4, 5, 6\} = \{1, 3\}$.
- **Combinations** of intersection, union, and difference can be used. When the union of C_a and E_a is selected, and the difference of this union and D_a is calculated, items 3 and 6 are retrieved: $(C_a \cup E_a) \backslash D_a = (\{3\} \cup \{5, 6\}) \backslash \{4, 5\} = \{3, 5, 6\} \backslash \{4, 5\} = \{3, 6\}$.

3.1.6 Conclusions

All ontologies describe and relate entities in different ways, but still the boundaries of the description power of any ontology remains the same. No matter what kind of formal

language is used in constructing an ontology or creating queries, the triadic pattern of Firstness, Secondness, and Thirdness can be used to explain it.

Category tree is a very clear model for depicting the top-most distinctions of ontologies. The principle of category tree can be intuitively understood by humans, which is why ontologists have used it for centuries (Appendix C), and an increasing amount of people are using the principle all around the world in building frame-based ontologies, and in retrieving information from the Semantic Web.

3.2 Formal Ontologies

Today's modern formal ontologies that are used in computer science are examined in this section. Section 3.2.1 discusses different sorts of formal ontologies and frameworks and section 3.2.2 examines in detail the standard Semantic Web ontology languages that will be used in the case example in section 4.

Ontologies were taken in use in computer science to facilitate knowledge sharing and reuse. Since the beginning of the nineties, ontologies have become more and more popular research topic investigated by several AI research communities including knowledge engineering, natural-language processing, and knowledge representation. More recently, the notion of ontology is also becoming widespread in fields such as intelligent information integration, cooperative information systems, information retrieval, electronic commerce, and knowledge management. The reason ontologies are so popular is in large part due to what they promise and give: a shared and common understanding of a domain that can be communicated between people and application systems [40].

Maybe the most popular definition of modern formal ontology is given in [63]: *An ontology is a formal, explicit specification of a shared conceptualization.* A *conceptualization* of some phenomenon in the world identifies and determines the relevant concepts and the relations of that phenomenon. *Explicit* means that the type of the used concepts and constraints are explicitly defined, i.e., they suit for describing also other phenomenons of the same kind and are not constrained to some single 'real' phenomenon. *Formal* refers to the fact that the ontology should be machine readable. Hereby different degrees of formality are possible. The meaning of 'machine readable' cannot be clearly specified, but a formal ontology has to be stored in a digital format. *Shared* reflects the notion that the ontology is not restricted to some individual, but accepted by a larger group. At the best, the group can include all the people and programs in the world.

Basically, the role of ontologies in knowledge engineering and in software engineering is to facilitate the construction of a domain model. An ontology provides a vocabulary or terms and relations with which to model the domain. Because ontologies aim at consensual domain knowledge their development is often a cooperative process involving different people possibly at different locations. People who agree to accept an ontology are said to *commit* themselves to that ontology.

3.2.1 Ontology Types and Frameworks

Depending on their generality level different types of ontologies may be identified that fulfill different roles in the process of building a knowledge-based system [58, 67]. In practice, any information structure can be called 'an ontology': the table of contents or introduction of this thesis, a back cover text of a book and object oriented or other kinds of databases can all be taken as ontologies, metadata, or category tree structures. There is not a widely accepted classification of ontology types, but some ontology types can be distinguished among others. The following classification is an example of an ontology itself: ontology of ontology types where one ontology can belong to more than one category[13]:

- **Representational ontologies** and **ontology frameworks** provide representational primitives without committing to any particular domain. These kind of ontologies do not express the exact purpose of the primitives, but only offer a framework that enables the usage of the provided representational primitives. Well-known representational ontologies are for example the Frame Ontology [63], and Resource Description Framework Schema (sect. 3.2.2). Class structures are the most common examples of representational ontologies. Every person who uses a computer stores information in a class or folder structure. Also set theory, predicate logic, and numerous other mathematical formalism can be taken as representational ontologies.

- **Top level**, **upper**, **generic, general, core**, and **common-sense ontologies** aim at capturing human common-sense knowledge about everyday life, providing basic notions about concepts like time, space, state, event, etc. [142, 51]. As a consequence, they are valid across several domains and provide a basic, domain independent vocabulary and object specifications to be used as the basis of other, more domain specific ontologies. Standard Upper Ontology [145] work group of IEEE has been specifying a standard top level ontology since 2001. The objective of the working group is to assist the development of ontologies and advance knowledge engineering in general by providing a common ground for more specific domain ontologies [113].

- **Metadata ontologies** like Dublin Core [154, 28] provide a vocabulary or categories for describing the contents of on-line information resources in the Web.

- **Domain ontologies** describe a reusable vocabulary of a given domain. Top level ontologies may be used as the foundation of a domain-specific ontology. Domain ontologies can describe domains around happenings such as surgery, ice-hockey, wedding, promotion (sect. 4.1), etc.

- **Application, Method and task-specific ontologies** Application ontologies specify the vocabulary required to model a certain application, and provide the base-structure of an application [18]. Application ontologies and domain ontologies can be very similar in nature. Task-specific ontologies provide vocabulary and knowledge used to solve problems associated to a certain task. Method ontologies provide the terms and knowledge to particular problem solving methods (PSMs). Domain-task ontologies are task-specific ontologies where the intended problem solving covers problems only in a given domain area [41, 140].

[13]This classification is based on [40] and on the sources that appear in the classification.

Part of the research on ontologies is concerned with envisioning and building technology that enables a large-scale reuse of ontologies at a world-wide level [121]. In order to enable as much reuse as possible, ontologies should be modular, and the modules should have a high internal coherence. This requirement among others is expressed in design principles for ontologies [62, 57, 150]. Assuming that the world is full of well-designed modular ontologies, constructing a new ontology is a matter of assembling it from the existing ones. For example, the *Scalable Knowledge Composition* project [82] aims at developing an algebra for systematically composing ontologies from already existing ones. It offers union, intersection, and difference as the basic operations. However, combining two ontologies is not so easily done as it may sound. There would be many problems and questions in creating a union even with two ontologies written in the same language. There are so many different languages and intended usages of ontologies that there is a long way to go before the information in many different kinds of ontologies can be used efficiently, and includes conversion between languages.

Ontolingua [38, 115] server provides different kinds of operations for combining ontologies: inclusion, restriction, and polymorphic refinement. Inclusion of one ontology with another has the effect that the composed ontology consists of the union of the two ontologies (their classes, relations, axioms).

The *SENSUS* system [147] provides a means for constructing a domain specific ontology from given common sense ontologies. The basic idea is to use so-called seed elements, which represent the most important domain concepts for identifying the relevant parts of a top-level ontology. The selected parts are then used as starting points for extending the ontology with further domain specific concepts.

Knowledge Interchange Format KIF [87, 88, 53] is a language designed for use in the interchange of knowledge among disparate computer systems, possibly created by different programmers at different times in different languages, and so forth.

The *Foundation for Intelligent Physical Agents FIPA* [42] has defined ontologies to facilitate communication of agents: common norms are needed to enable a reliable commercial activity for example.

Three of today's largest formal ontologies are briefly discussed in appendix C, and the standard Semantic Web ontology languages are discussed next.

3.2.2 Semantic Web Standards

All standard languages explained in the proceeding are formal recommendations of the World Wide Web Consortium W3C [158]. XML-based languages [20, 8] play a major role in describing on-line information on the Internet. Resource Description Framework RDF [95, 127, 128], an application of XML, is a general framework for describing any resources reachable through the Internet. An RDF description can include the authors of the resource, date of creation or updating, the organization of the pages on a site, information that describes content in terms of audience or content rating, key words for search engine data collection, subject categories, and so forth. RDF enables sharing information on Web sites and helps software developers to build products that can use the RDF-descriptions to provide better search engines and directories, to act as intelligent agents, and to give Web users more control of what they are viewing. As explained in section 3.1.4, the principle of triads is applied in the very basis of RDF. In RDF

information is expressed in form of triples: `Subject-Property-Value`, of which all can be URI's.[14]

```
Subject:  http://www.example.org/foo.html
Property: http://purl.org/dc/elements/1.1/creator
Value:    http://www.example.org/staffid/85740
```

`Subject` is a Web page that has values for a set of one or more properties, of which `Property creator`[15] is under scope. The `Value` of `creator` is `http://www.example.org/staffid/85740`. It is not important what the triples are called, but as with Firstness, Secondness, and Thirdness, `Value` is the intersection of `Subject` and `Property`. `Property creator` is one aspect of `Subject`, and `Value` is what is common to both `Subject` and `Property`. Also other query languages can be explained with triads: with SQL the clause `select X from Y` selects all values of `X` with the given `Y`[16].

The simplest *Frame-based* Semantic Web ontology language is RDF Schema [21, 69]. RDFS is used to construct the promotion-ontology in section 4.1 and is therefore examined more thoroughly than other languages. The table below clarifies concepts used with object-oriented programming languages and frame-based ontology languages.

Programming languages	**Ontology languages**
software development	ontologization
programmer	ontologist
class	class, category, frame
variable	property, attribute, slot, field
object	instance, individual
value of variable	value of property, attribute, slot, field
type of variable	range or facet of property, attribute, slot, field
class that contains the variable	domain of property, attribute, slot, field

The generic concepts of the RDFS vocabulary are called *classes*. Classes are organized into conceptual hierarchies, just as explained in section 3.1. For example, class **Place** represents the generic category of classes **Building**, **Park**, and **Forest**. **Building** is a superclass of classes **Apartment building** and **Train station**.

Classes may have *properties*. Successors inherit properties of their predecessors, and if **Building** has an *architect*, also **Apartment building** has an architect. **Apartment building** can naturally have extra properties in addition to the inherited ones. Properties have constraints that state what kind of values can a certain property have. Such features are sometimes called 'facets'. Also *instances* can be created. Each instance belongs to one or more classes and can have values for all the properties defined for its classes, but the values can also be left unspecified.

[14] URI=Uniform Resource Identifier. The generic set of all names/addresses that refer to resources. URL, Uniform Resource Locator is a subset of URI, an informal term associated with popular URI schemes: http, ftp, mailto, etc. URI's cannot contain empty spaces among other restrictions [149].

[15] `creator` is used as an abbreviation of http://purl.org/dc/elements/1.1/creator

[16] An RDF triple is logically equivalent to an SQL [61] table that has for example one row and two columns. However, RDF is a more flexible basis for Web ontology languages because there is no need for explicit tables or public and foreign key specifications et cetera, and it is easier to share RDF databases in the Web as plain text files because every RDF statement is a individual, which has URI pointers to other statements.

Figure 14 depicts an RDF Schema that corresponds to $\top$ > **Place** > **Building** > **Finlandia-House**. The RDF statements in the figure are examined in more detail in

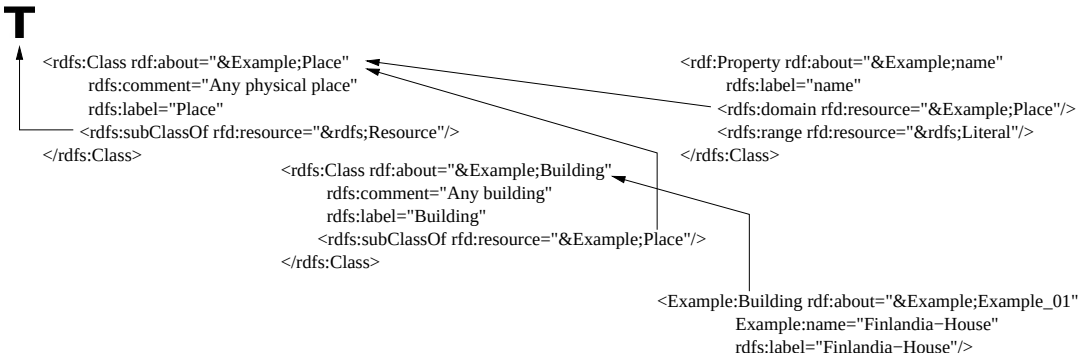


Figure 14: A visualization of RDFS Class-Property-Instance relationship, where the arrow points to the resource that is referenced by the source of the arrow.

the following. Character strings that start with '&' are abbreviations (or macros) of accurate namespace URIs that are situated in the beginning of the files that contain the statements. RDF statements follow the XML syntax [127]:

```
<rdfs:Class rdf:about="&Example;Place"
         rdfs:comment="Any physical place"
         rdfs:label="Place">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
```

`rdfs:Class` indicates that this is a class-description, i.e., the type of this resource is `rdfs:Class`. `rdf:about="&Example;Place"` indicates that `&Example` is the URI of the RDF Schema where the class is defined, and the exact identifier of this class is `&Example;Place`. As stated earlier, `&Example` is an abbreviation, when the full URI could be something like `http://www.MySchemas.com/Example`. `rdfs:comment="Any physical place"` is used to provide a human-readable description of a resource. `rdfs:label="Place"` is used to provide a human-readable version of a resource name and indicates that character String `Place` is the `rdfs:label` of this class. `rdfs:subClassOf rdf:resource="&rdfs;Resource"` indicates that this class is a subclass of `Resource`, the highest abstraction level of RDFS that corresponds the universal type $\top$. Top level class **Place** has a subclass:

```
<rdfs:Class rdf:about="&Example;Building"
         rdfs:comment="Any building"
         rdfs:label="Building">
        <rdfs:subClassOf rdf:resource="&Example;Place"/>
</rdfs:Class>
```

`rdfs:subClassOf rdf:resource= "&Example;Place` indicates that class **Building** is a subclass of class **Place**, and **Building** inherits all the properties of class **Place**.

Properties are defined in a similar fashion than classes:

```
<rdf:Property rdf:about="&Example;name"
         rdfs:label="name">
        <rdfs:domain rdf:resource="&Example;Place"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

`rdf:Property` states that this is a property specification, `rdf:about= "&Example;name"` gives an ID to the property, and `name` is the `rdfs:label` of the property. `rdfs:domain rdf:resource= "&Example;Place"` states that class `Place` and all of its successors have property `name`. `rdfs:range rdf:resource= "&rdfs;Literal"` states that the value-type of this property is `Literal`, i.e., a character String. Properties can also exist without belonging to any class, when the property has no domain specifications. Also sub-properties can be created with RDFS, which are used to specify that one property is a specialization of another. Instances can have values for the the properties specified for their parent classes, or inherited from the parent classes' predecessors:

```
<Example:Building rdf:about="&Example;Example_01"
         Example:name="Finlandia-House"
         rdfs:label="Finlandia-House"/>
```

`Example:Building` states that the type of this instance is class **Building**, and the name of the RDF Schema where the class specification can be found is `Example`. `rdf:about="&Example;Example_01"` is the unique identifier of this instance. `Example:name="Finlandia-House"` states that the value of this instance's property `name` is "Finlandia-House". `rdfs:label="Finlandia-House"` again gives a human-readable label for this instance.

Like with RDF, also RDFS is used by creating queries in triple-form, just as explained is section 3.1.4. The goal in the below example is to retrieve a list of URI's of all subclasses of $\top$ of the schema in figure 14. In RDQL the queries are represented in form *subject*, *object*, and *predicate*[17]:

```
SELECT ?x
where (?x,rdfs:subClassOf,rdfs:Resource)
```

The result `x` is the ID of class **Place**: `&Example;Place`.

[17]Abbreviations are used. RDQL tutorials are available [130].

RDFS provides simple modeling primitives that can be used in constructing very simple ontologies but more specific ontologies should be created with other languages. There have been several efforts to create languages that accommodate description logics and have clear rules for semantics in using frame-based modeling primitives. Most recent of them are *Web Ontology Language OWL* [117, 107] and *Semantic Web Rule Language SWRL* [148] that can handle multiple range, cardinality and other specifications of properties, total or partial equivalence and disjointness of classes (property inheritance), and handling different sorts of sequences and collections. OIL [114] and DAML [33] can be taken as forerunners of OWL and SWRL.

As these new languages evolve, RDFS might be totally forgotten as it is now since new languages with greater descriptive power can be used to create also simple ontologies. It is possible to automatically modify an RDFS ontology into a more complex one, but doing the process vice versa includes a high degree of specification of additional rules, for example in expressing sequences, and the process would be very impractical.

RDFS has a model theory [128, 95], but the semantics of RDFS or any other ontology language that should have domain-independent means to describe things will always remain more of less unspecified. The semantics tells *how* the language should be used, and specification of the semantics is essential to any language paradigm because clear semantics highly facilitates the creation of homogeneous ontologies that can be efficiently understood and reused across domain boundaries.

There are no clear rules about how classes, instances, and properties of RDFS should be used. Instances can have sub-instances as well as classes can have subclasses, and so category structures could be created by using only classes or only instances, or even only properties that can be defined for both classes and instances. Also instances, classes, and even properties themselves can be values of properties.

Ontology languages are *incomplete* means to explain relations of things, and irrational statements can be declared that follow the syntax and semantics of a language. According to Gödel's *first incompleteness theorem* (1931) [144], within any mathematical language that can be applied to natural numbers, or with which can we can perform operations on natural numbers, are statements that cannot be proven to be either true or false. This indicates that there is no sense in trying to create a 'perfect' ontology language because there cannot be one. The semantics of ontology languages will always be more or less unclear because it is impossible to define deterministic guidelines about how to describe all things in general.

Whatever means are chosen to create ontologies, it is always good to remember not to possess plurality without necessity, and to keep the systems open for possible future needs. Ontologies created with XML and RDF-based languages are surely easier to keep open than others since the syntax is commonly agreed and there is a growing quantity of parsers, editors, and other tools that can handle these. RDF is a very flexible and a good basis for ontology languages because all types of entities can be represented and queried in a similar fashion, which is not the case with traditional database query languages like SQL. The prevalence of frame-based ontology editors and their popularity among users suggests that the frame-based paradigm will prevail in the field of ontology development.

4 Case Study on Ontologization, Annotation, and Retrieval

This section goes through creating and using a formal ontology in practice. The section is balanced between an individual case example and the general side of the topic, i.e., there are theoretical parts and case-dependent parts within the text. Other projects of the same nature have been executed and the results have been promising [134, 155, 99, 35, 123, 73].

The process of creating a domain ontology is examined first in section 4.1. The created ontology is used in annotating images in section 4.2, and in section 4.3 the ontology and the annotations are used with an image exhibition software. The English terms used with the ontology and with the exhibition's user interface are translated from Finnish.

4.1 Ontologization

> *Occam's Razor*, the most important guideline of ontologization has been named after William of Ockham (c. 1285-1349), and gets easily forgotten during the ontologization process: *"Plurality should not be posited without necessity"*. The ontologist should not use more concepts than are needed for the cause of the ontologization effort. The amount or possible relations and confusions rises exponentially along the amount of used concepts, just as in real life: person X can check a friend's phone number from a telephone catalog, ask it from the operator, check it from the memory of a mobile telephone, Email a friend to ask it, who could again call another friend who could again call X to ask it.

The case is to ontologize a traditional happening of Helsinki University called *Promotion*. Promotion stands for the process where graduated masters, doctors, and other *promovends* are promoted to their titles. There have been promotions since year 1643 in Finland. The tradition of Finnish promotions was copied from Uppsala University in Sweden where it came from France dating back to 13th century. Today about all of the Finnish universities that have promotions use more or less the same procedures as in Turku almost 400 years ago. Different schools all around the world have the same kinds of traditions with their graduation ceremonies: the promotor admits a hat and a diploma to the graduated students.

4.1.1 Requirement Analysis

Ontology development has similar characteristics to software development in general[18]. After an overall project plan a software development project usually proceeds to requirement analysis. Requirement analysis should specify all that the client requires from the final software product, and in many cases only those strict requirements. The requirements of the client are defined without taking a stand to the software or ontology itself. The requirements specify what should be done, and the design specifies how it is done [65]. In ontology creation, and as usually, this stage of the process is especially important. Domain ontologies expand very easily out of the domain borders, which in the end

[18]The terms used with traditional programming languages and frame-based ontology languages were described in the graph in p. 61.

leads to unnecessary and time-consuming ontologization of the world. By staying within borders of the requirement analysis the ontologist can concentrate on the essential problems and not consume time on thinking about what should be included in or excluded from the ontology.

In this case the client requirements were vague: "construct a digital photograph exhibition software about the promotion happening with a set of 600 photos." The main factors that guided the ontologization were 1) the end users' common-sense requirements, 2) requirements of the client, 3) requirements of the application programmers, and 4) common-sense and experience of the ontologist and domain experts. The ontologist can be seen as a medium who constructs the ontology objectively according to requirements of the other participants. In reality the change of ontologist affects the ontology. There is not *one and only right* solution to most of the problems in ontologization process and there is always a pay-off in choosing the 'less bad' choice. Every person has a slightly different model of reality and a joint ontologization process is an extreme example of a situation where opinions of people can collide. When there is a problem about an opinion with two of the participants it is useful again to remember the requirements and needs of the other participants to solve the problem with a mature debate. A good ontologist has knowledge of modern tools and principles and can take into account the requirements of other participants without problems.

The ontology's life cycle specification is also important. If it is sure that the ontology will be used with different systems for a long period of time, then the documentation about the ontology's structure should be very exhaustive. With very simple ontologies that are used only once the documentation might not be needed at all. In this case the ontology's structure changed so often that a literal documentation was decided to be just adequate for the programmers to use.

As a data storage an ontology contains structured information. The structure itself can be formed in many ways and there is a fuzzy line between needful and unnecessary paths of finding information even though the information would remain identical. In some cases even the existence of one very long possible path is dangerous and can cause problems in application programming, as usually with cyclic nets. So, the overall goal in ontologization, disregarding the domain, is to construct an information structure with only the necessary information in a compact form where all user groups can find every bit of information as quickly and as easily as possible. In this case the ontology should be able to describe all photos about promotion ceremonies taken place in the past or in the future, and contain information about the history and the present state of the ceremonies.

4.1.2 Languages and Tools

RDF Schema (sect. 3.2.2) was selected as the implementation language of the ontology. There was no need for a more descriptive language and the domain could be ontologized up to an adequate degree with RDFS. The fact that RDFS was and is a W3C standard also made it a good choice. A variety of ontology editors that can import and export RDFS are currently available [29] but when this project started in the beginning of year 2002, Protégé-2000 [124, 61, 126, 36] was considered as the most reliable choice and could be taken easily in use. However, Protégé does not support multi-instantiation even today, i.e., one instance can belong to one class only even though RDFS specification allows multiple `rdf:type`'s for one instance. Protégé supports only multiple inheritance of

classes, which is why classes were used instead of instances whenever multiple inheritance was needed.

To use RDFS ontologies with applications, methods for creating queries and mapping the results of the queries with a user interface are needed. Jena [84], a Java-based [83] toolkit for creating Semantic Web applications that supports RDFS was used. RDQL [130] (p. 27), a query language for RDF files comes with Jena but Jena can be used also directly in an object-oriented fashion to create queries without RDQL. Perl [120] was used for transforming and doing additional fixes to the files containing the ontology in a textual format. Changing something with an ontology editor might be slower than doing the same thing with scripts. The typical find-replace procedures can be done with normal text-editors but complex editing problems must be handled with scripts[19]. W3C RDF validation service [129] proved also useful as well as RDF visualization tools [52, 131].

4.1.3 Prototype Process Model

The requirement analysis was unclear and new technology was being used in the project. The ontologization could not be divided into design and implementation stages, and *prototype model*[20] [65, 101] was the only suitable process type. Usually the clients of a software development project are concerned mainly with the looks of the user interface, it being the part of the software that the end users see. In this case however the ontology's category structure was a fundamental part of the exhibition software's information representation structure and the clients participated in the ontologization. Figure 15 depicts a diagram of the used prototype model.

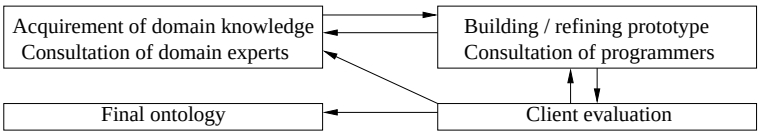


Figure 15: The development model used in this ontologization process.

The process starts from acquirement of domain knowledge, possibly with the aid of domain experts. When new information is received the ontologist can build a prototype, possibly consulting the programmers. The prototype is then evaluated by the client to decide how to refine it. If client evaluation is not needed the ontologist can proceed by gathering more information that is added into the ontology. The ontology can be refined also according to the client's feedback without acquiring domain knowledge from anywhere else.

The process goes on until the ontology is accepted by the client. With more complex ontologies the client acceptance is only a matter of how the information in the ontology is represented to the client but with frame-based ontologies the category structure can be clearly understood by a client who is not a software expert.

The first prototypical version of the ontology was constructed in a couple of months using textual summaries about the domain and some additional explanations from the

[19]The following kinds of regular expressions can be specified with Perl: "If there is a certain character string X somewhere after <, and before the next > in a file, then replace X with another string Y".

[20]Or the Evolutionary model.

client. A prototype of the exhibition software that used the prototypical ontology was also constructed. The prototype was introduced to the clients who accepted the main principles to be applied with the larger work.

4.1.4 Acquiring the Domain Knowledge

In the beginning of ontologization the ontologist does not have to think about the structure of the ontology itself but has to get acquainted with the subject domain. It is important first to understand the domain before starting to create a formal ontology. If categories and their relations are created before understanding the domain it is probable that logically wrong or not optimal choices are made, which leads to unnecessary and time-consuming extra repairing work later on. It is eventually less time consuming to use time on finding a vast amount of source material (literal, visual, audial) in the first place, than to always start again the process of searching for the literature when some unexpected information is needed.

To make the ontology efficiently usable, reusable, and accepted by the largest possible group of users, the ontologist should search for and use the possibly existing vocabulary specifications and other guidelines of the subject domain. The vocabulary of the domain has to be understood and accepted by domain experts, clients, users, and the programmers. The relieving thing here is that the category structure of concepts facilitates understanding the meaning of terms that might otherwise be hard to understand: *predecessors and successors of an arbitrary category* **X** *determine the meaning of* **X** (p. 15).

In this case, there was a short thesaurus that had been used with all the material annotated in the database of the Library of University of Helsinki. However, the thesaurus contained only a fraction of the words that were needed to describe the domain and most of the used terms were picked from the literature by the ontologist and affirmed by domain experts and clients.

The material used in the ontologization was based on historical literature [92, 90, 9], modern literature [66, 109, 94], some Internet sources [68], client recommendations, and on commemorative material about a few certain promotions [43, 44, 45]. Also the set of 628 images with image texts that served as the visual content of the exhibition guided the process. Seeing what would be confronted in the annotation stage made the ontologist to create the ontology accordingly.

Several meetings were held with the domain experts who had participated to and arranged promotions. They reviewed the ontology and gave hand-to-hand knowledge up to a degree when some questions were only a matter of opinion between the domain experts. In these cases the ontologist fused the information into a cohesive whole. However, in the end the client had the final word and the agreements made with the programmers (such as the annotation structure and top level classes) had to be maintained because changes in the ontology's structure would have caused major malfunctions in the exhibition software. By frequently consulting all the participants the ontologist managed to get all the needed information into the ontology and the ontology also got accepted by all the participants.

4.1.5 Selecting Top Level Classes

The process of constructing the actual formal ontology in RDFS can be started by selecting the top level classes. These classes answer to question "What has to be described?" while the rest of the ontology answers to "How is it described?" It must be understood that one top level class alone does not provide the needed information to describe a single photograph, but all the top level classes function as a whole as means to annotate and retrieve the photos. The top level classes have to have the greatest contrasts to keep them from representing redundant information. An overall description of the domain can serve as the source material in this stage. The selection is done in a two-step process:

- **1. Picking up classes**. Write down or underline the meaningful things that clearly come out of the representation and are repeated in the text more than others[21], excluding those that are already included, if any.
- **2. Occam's Razor.** When you have identified some classes, look at them together. Use the principle of triads and the has-test to create a hierarchy of these classes (sect. 3.1.4). Move back to step 1 until you cannot find new classes from the presentation, or when the new classes suit only as subclasses of the already included ones.

After repeating the steps 1 and 2, the overall top level class structure should have been formed. The top level classes are depicted in figure 16. All the top level classes are

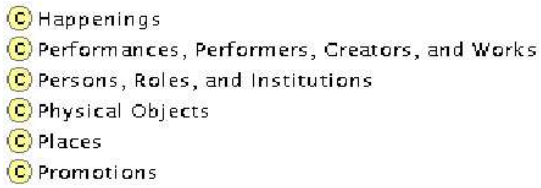


Figure 16: Promotion ontology's top level classes.

in contrast with each other and describe different sorts of things relative to each other. **Performances, Performers, Creators, and Works** is an exception because it shares some successors of **Persons, Roles, and Institutions** and **Happenings**: performances are also happenings and performers have a role. The other top level classes do not share successors. **Places** collects the kinds of places that have a street address. **Places** could be taken also as **Physical objects**, but in contrast **Physical Objects** collects touchable objects that are smaller in size than those collected by **Places**, such as paintings, badges, and rings that do not have a street address. Also **Persons** can be taken as **Physical Objects**, but in contrast **Physical Objects** does not collect living beings. **Promotions** collects information about the general characteristics of different promotions, such as the date of the conferring ceremonies and the essential persons and their roles. **Promotions** could be taken as a successor of **Happenings**, but **Happenings** collects only happenings that occur during promotions in a hierarchy.

The obligatory requirement of the present class structure is that it should cover the subject domain so that every possible photograph about the domain could be placed

[21]This technique is used with traditional software design.

below one or more top level classes. If this can be done, it is only a matter of deepening the structure to make it more descriptive. If the top level classes do not cover the domain, more top level classes have to be created.

4.1.6 Working up the Ontology

All classes of the ontology represent the unchanging continuants (p. 18), the stable structure of things relative to the promotion happening. A part of the instances are also continuants, like those instances that describe the faculties and student clubs. Another part of the instances describes the occurrents that are different in every promotion, such as the persons who participate to promotions in different roles. The borderline between the continuants and occurrents is however vague: the promotor changes in every promotion, but the same principal usually participates to more than one promotions. The top level classes of the ontology do not have direct instances, but have more specific successors that do have instances. If the class-instance structure is seen as a tree, the instances are situated approximately as the leaves of the tree.

In the optimal case, once the classes have been selected the overall hierarchy is ready and only the properties of the classes have to be created before creating the instances. In practice the class structure, properties, and their value specifications change during the process. Even with nearly perfect design methods something usually has to be reworked. Naturally, the structure of the annotation schema affected greatly on the ontology's structure. The annotation is discussed in detail in section 4.2 and for now it is enough to understand that the annotation is done by linking the classes and instances of the ontology to photographs, in a similar fashion than in section 3.1.5. The properties (`rdf:Property`) serve as the means to identify the instances in the annotation process and are used in semantic reasoning (sect. 4.3).

The lacking of tools to implement partial inheritance of properties enforced the ontologist to create a minimal set of properties: a class inherits all properties of all its predecessors, of which many can be useless for describing an instance, and giving values for all of the properties can include redundancy in some occasions. One one hand the lacking of partial inheritance is a good feature: having very many properties requires more organizational efforts from the ontologist, and it is good to keep the ontology minimal. On the other hand the partial inheritance would allow richer expressiveness, and if planned accurately, would decrease the redundancy.

The lacking of multiple inheritance of instances that was due to the ontology editor caused problems, and in many cases the things that would have otherwise been described with instances were described with classes to be able to place them under more than one class.

Creating new classes and properties walk hand in hand: does a property suit for a class or does a class suit for a property? Top level class **Places** serves as an example of deepening the class structure. It has to describe all places used in promotions up to an adequate degree. The class is given three properties that come first in mind: *name of place*, *address*, and *description of place*. The properties' value type is literal. The ontology should be as simple as possible, which means that a property should suit for as many classes as possible. However, *name of place* does not suit for any other top level classes, no more than *description of place*. A property's `rdfs:label` should not bind the property to a certain class only if it can be avoided. If it would, there would eventually

be very many properties such as *name of happening* and *name of physical object*, etc. To avoid this the `rdfs:label`s of properties are modified: *name of place* is changed into *label* and *description of place* into *description*. Only *address* is left unchanged since it does not need to belong to other classes than **Places**.

Having the mentioned minimal properties in use, a total of 47 instances (fig. 17) are created for **Places** based on commemorative material [43, 44, 45].

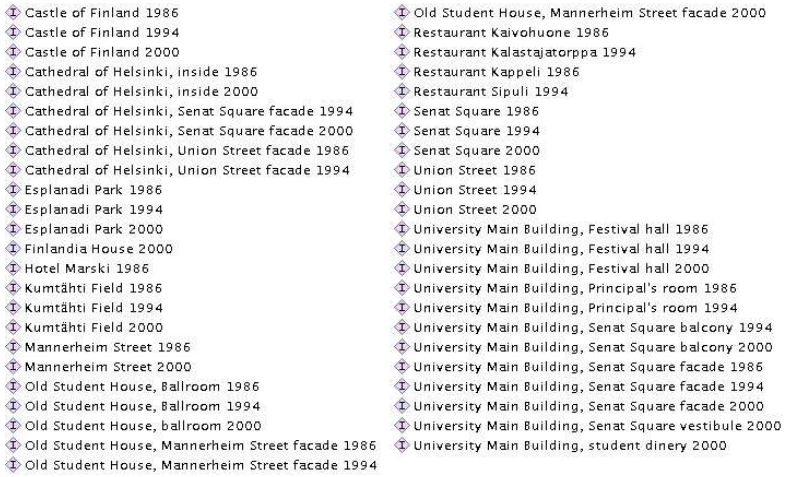


Figure 17: Instances of **Places**. The value of property *label* is shown.

The `rdfs:label`s of the instances fit on one screen, and any user could find the wanted ones from within the others, but the final exhibition would have a lot of more instances and searching would then require browsing up and down in the user interface. This is why there is now a need to deepen the class structure and create subclasses for **Places**. The instances of figure 17 describe buildings, streets, squares, and an island, and subclasses are created for **Places** that describe all these. When both classes and instances are thought as categories, the principle of triads and has-test can be used again. The instances of figure 17 are moved under the new classes in figure 18.

Figure 18: Refined subclass structure of **Places**. On the right side of a class is an integer indicating the amount of its instances.

There are still 30 instances under **Buildings** in figure 18, which causes a need to further deepen the structure.

The final structure of **Places** is depicted in figure 19, where the instances of **Buildings** of figure 18 are distributed between the subclasses of **Buildings** according to their characteristics.

Figure 19: On the left side of the classes that have subclasses is a ball-like symbol with a line pointing either down or right. Right-position indicates that the subclass- structure is hidden, and down-position indicates it is revealed.

Some **Buildings** are traditionally used in every promotion like **University Main Building**, **Old Student House**, and **Cathedral of Helsinki**, which were given their own classes. There are photos taken of places from different directions, which is why instances describe different visual views of the places. In figure 19 the `rdfs:label`s of the 6 instances below **Cathedral of Helsinki** were changed, resulting in 3 instances. There was no longer a need to tell that the instances describe the Cathedral of Helsinki with the `rdfs:label`. Any user could understand it because the instances are under the class that tells this information. Other information that was previously included with the instances' `rdfs:label`s could be described with successors of the other top level classes: the year was discarded from the end of the `rdfs:label`s because the information about the time comes by linking an instance of successor of **Promotions and Festive Sessions** (p. 39) with an image in the annotation stage. After the year was discarded some of the `rdfs:label`s turned out to be identical, like the two instances with `rdfs:label` 'Senate Square facade'. In these kinds of cases only one instance was maintained.

Now the class structure has been extended without adding more properties to **Places** or to its successors. In the final annotation stage more instances were created, but as assumed, they were centered on few specific places. The same kind of principle was used with all the top level classes.

The previous example also concretized the usefulness of the category tree approach. In figure 17 the amount of characters[22] that were used to describe the Cathedral of Helsinki was 256. In figure 19 the amount was only 67[23]. And further, the annotator does not even have to write any text, but can only select instances from the ontology. The more images are annotated, the greater is the relative benefit.

4.1.7 Testing the Ontology

Before the testing stage the actual 628 photographs that were to be annotated did not directly affect on the structure of the ontology. They however served as a perfect material in testing the ontology. If the ontology is not tested before the actual annotation starts, unpleasant changes might have to be made to the ontology during the annotation process. The changes might make the previously made annotations partly or totally useless and

[22]The total amount of characters in the instances' `rdfs:label`s.

[23]When the amount of characters in the `rdfs:label` of class **Cathedral of Helsinki** is also counted.

the annotator/ontologist would then have to keep a memo of subsequent annotations and changes, which could easily lead to an unsolvable mess with all the classes, instances, images, and their relations.

During the testing many unexpected things had to be described, which again raised a need to rework the ontology, mainly by adding new classes, instances, and properties, but also by removing or combining the useless ones. The testing itself is straightforward. Images are annotated and two things are observed (categories again denote both classes and instances):

- Can the present category hierarchy describe the photographs' *physical elements* (sect. 1.2) accurately enough? One category alone should not describe too wide concepts or too many images. If this is the case the hierarchy should to be extended.
- Is the hierarchy too precise? When some categories are not used at all they should be deleted or combined with other categories. When two or more categories describe very similar things they should also be combined into one, following the triadic principle (sect. 3.1.4).

It has to be analyzed somehow just how deep and descriptive the category structure should be or if it is already good enough, keeping in mind the requirement to keep the structure minimal. If there are as many categories as photos it is quite clear that the structure is too deep. If there is only one category and hundreds of photos the structure is too shallow. The answer lies somewhere in between.

One way to analyze the soundness of the structure is to measure the *relative change*. The amount of categories is relative to the amount of photos they describe. A certain percentage x of the photos can be annotated, reforming the category structure to be optimal for describing these. Then another $x\%$ of the photos is annotated. If the structure has to be changed as much as it changed with the first $x\%$, then something is wrong with the ontology, or the photos are very heterogeneous. If the structure changes less than with the first $x\%$ the ontologist is probably on the right course.

Let $\Delta\mathcal{I}$ denote the amount of new images and $\Delta\mathcal{C}$ the amount of new categories. In the general case, in the beginning of ontologization the amount of categories can rise almost linearly relative to the amount of annotated images: $\Delta\mathcal{I} \approx \Delta\mathcal{C}$. When $\Delta\mathcal{I}$ is big enough it affects $\Delta\mathcal{C}$ only to a logarithmic degree: $\Delta\mathcal{C} \approx log(\Delta\mathcal{I})$. This is quite natural: the more images are annotated the more words or categories are needed to describe them, but no matter how many images are annotated they all can be described with a finite set of words or categories.

4.1.8 Analysis of the Created Ontology

The ontology's final formation is analyzed one top level class at a time in the following, apart from **Places** that has already been analyzed. Some conventions have to be used in order to clarify the explanation of classes, properties, and instances:

- The references to the RDFS specifications such as `rdfs:label` and `rdfs:range` are typed in `typewriter font`, as well as some specifications of the ontology editor.
- The `rdfs:label`s of classes are typed **in bold font** and those of properties in *italics*.

- When term 'category' is used it can denote both classes and instances, but not properties.
- The following phrases mean that **X** is a subcategory of **Y**: "**X** is placed under **Y**", "**X** is created for **Y**", "**Y** has **X**".
- Let the meaning of the following phrase be clarified: "successors of **X** are values of successors of **Y**." Classes **X** and **Y** and their successor classes can all have instances. **Y** can have or can have inherited properties from its predecessors. One or more of these properties take instances of class **X**, and instances of **X**'s successor classes as values. Also, when "property *p* takes successors of **X** as values", the *p* can have all instances of **X** and all instances of all successor classes of **X** as values.
- The value of `rdfs:label` of instances is based on some `rdf:Property` defined for the instances' parent class. The value of instances' `rdfs:label` could not be asserted directly because of the ontology editor. For example, property *label* is not the same as `rdfs:label` that belongs to every class, property, and instance; only the value of `rdfs:label` can be set to be the value of property *label*[24].

Figure 20 explains the visual representation of classes and properties that is used from now on. A line is drawn between the properties and the class that the properties are defined for. The blue `S` on the left side of *person in role* indicates that the `rdfs:domain` of property *person in role* is class **Essential Roles**. White `S` indicates that the property is inherited from a predecessor class of **Essential Roles**. `Name` is the `rdfs:label` of

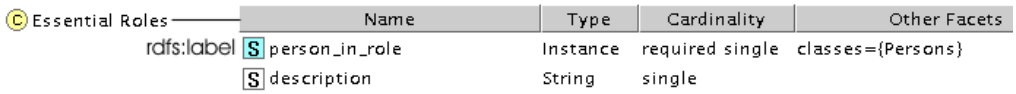


Figure 20: The style of representing classes and properties. The bottom dash _ in property name is due to the convention of the ontology editor.

the property. `Type` is the `rdfs:range` of a property, that is either `String` (=`Literal`) or `Instance` with all the top level classes and their successors. `Other Facets` indicates the `rdfs:range` of the property when the value type is `Instance`. In figure 20 the `rdfs:range` of property *person in role* is class **Persons**, and so *person in role* can have only successors of **Persons** as values. The `rdfs:label` that is on the left side of *person in role* indicates that the value that is given to *person in role* serves as the `rdfs:label` of instances of **Essential Roles**[25].

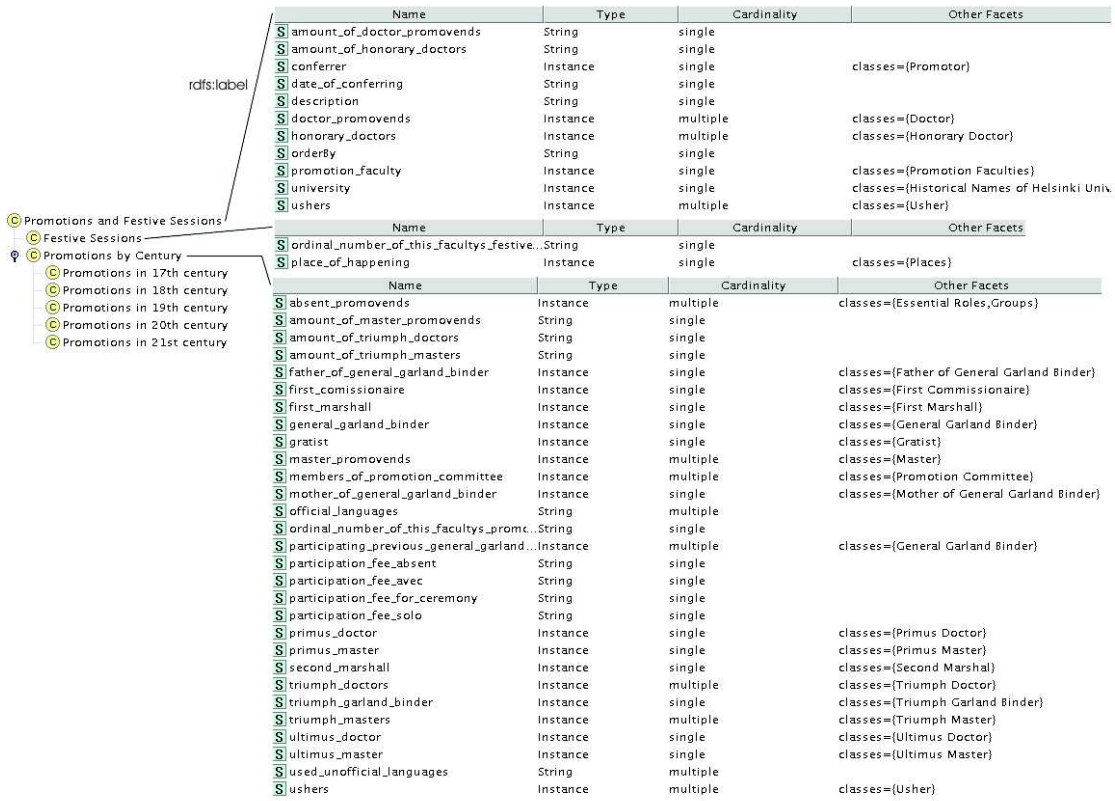
`Cardinality` is either single or multiple and indicates how many values a property can have, or has to have. This OIL/OWL (sect. 3.2.2) feature was not included in the final RDFS-ontology that was used in the exhibition, but was used occasionally in the ontologization and annotation stages. The same goes with `required` that specifies that this property must have a value, when without the explicit specification the value would not be obligatory.

[24]Value of property *name* was the value of the instance's `rdfs:label` in fig. 14 in p. 26.

[25]Property *person in role* takes successors of **Persons** (p. 40) as values. The `rdfs:label` of successors of **Persons** in based on property *name of person*, and so also the `rdfs:label` of successors of **Essential Roles** is based on the textual value of *name of person*.

Promotions and Festive Sessions

Every photograph is connected to a certain time and place. Even though in most of the cases the date (day, month, year) of the photographing process was known the needed accuracy of time specification is in the level of single promotions, represented by property *date of conferring* (fig. 21), that served also as the `rdfs:label` of the instances. It is more important to know that a happening occurred during **Garland binding day** (successor of **Happenings**) in a promotion arranged in year x than to know that it occurred in the 11th, 12th, or 13th of June for example.



Name	Type	Cardinality	Other Facets
amount_of_doctor_promovends	String	single	
amount_of_honorary_doctors	String	single	
conferrer	Instance	single	classes={Promotor}
date_of_conferring	String	single	
description	String	single	
doctor_promovends	Instance	multiple	classes={Doctor}
honorary_doctors	Instance	multiple	classes={Honorary Doctor}
orderBy	String	single	
promotion_faculty	Instance	single	classes={Promotion Faculties}
university	Instance	single	classes={Historical Names of Helsinki Univ
ushers	Instance	multiple	classes={Usher}

Name	Type	Cardinality	Other Facets
ordinal_number_of_this_facultys_festive...	String	single	
place_of_happening	Instance	single	classes={Places}

Name	Type	Cardinality	Other Facets
absent_promovends	Instance	multiple	classes={Essential Roles,Groups}
amount_of_master_promovends	String	single	
amount_of_triumph_doctors	String	single	
amount_of_triumph_masters	String	single	
father_of_general_garland_binder	Instance	single	classes={Father of General Garland Binder}
first_comissionaire	Instance	single	classes={First Commissionaire}
first_marshall	Instance	single	classes={First Marshall}
general_garland_binder	Instance	single	classes={General Garland Binder}
gratist	Instance	single	classes={Gratist}
master_promovends	Instance	multiple	classes={Master}
members_of_promotion_committee	Instance	multiple	classes={Promotion Committee}
mother_of_general_garland_binder	Instance	single	classes={Mother of General Garland Binder}
official_languages	String	multiple	
ordinal_number_of_this_facultys_promc...	String	single	
participating_previous_general_garland...	Instance	multiple	classes={General Garland Binder}
participation_fee_absent	String	single	
participation_fee_avec	String	single	
participation_fee_for_ceremony	String	single	
participation_fee_solo	String	single	
primus_doctor	Instance	single	classes={Primus Doctor}
primus_master	Instance	single	classes={Primus Master}
second_marshall	Instance	single	classes={Second Marshal}
triumph_doctors	Instance	multiple	classes={Triumph Doctor}
triumph_garland_binder	Instance	single	classes={Triumph Garland Binder}
triumph_masters	Instance	multiple	classes={Triumph Master}
ultimus_doctor	Instance	single	classes={Ultimus Doctor}
ultimus_master	Instance	single	classes={Ultimus Master}
used_unofficial_languages	String	multiple	
ushers	Instance	multiple	classes={Usher}

Figure 21: Structure of **Promotions and Festive Sessions.**

The timely ordering of **Promotions** goes first by centuries represented by classes. The classes have instances that describe single promotions that occurred during the corresponding century. **Festive Sessions** describes promotions where the promovends are promoted to their titles in a less ceremonial way. Every property of every instance was not given a value, but with all the instances at least *promotion faculty*, *university*, and *date of conferring* were given values. There can also be several promotion ceremonies during the same date, which raised the need to distinguish separate promotions from each other by other means. Also *promotor* and *general garland binder* were given values if these could be found from the used literature.

Most of the properties were not used in the exhibition but they were very useful during the ontologization and annotation processes. The instances describe essential information about single promotions and the ontologist could easily check out information using them. Without these instances the ontologist would have had to check out the information from books, papers, and other sources, which is much slower than to check it with a couple

of mouse clicks having the needed information present at all times when the ontology editor was open. Successors of **Promotions and Festive Sessions** were linked to about 95% of the images in the annotation, which is more than with any of the other top level classes.

Persons, Roles, and Institutions

Persons, Roles, and Institutions collects **Persons**, **Essential Roles**, **Groups**, and **Institutions** (fig. 22). There were altogether 507 persons annotated in the system. All names of the persons could not fit on one screen and the class structure needed to be extended to avoid up-down scrolling in the UI. Subclasses **A**, **B**, **C**, ...,**Ö** were created to describe persons according to the first letter of their family name. An alphabet does not suit as a subclass of **Persons** in a philosophical sense but it is very intentional and intuitively clear to find a person by the first letter of family name: *predecessors and successors of an arbitrary category* **X** *determine the meaning of* **X** (p. 15). To increase

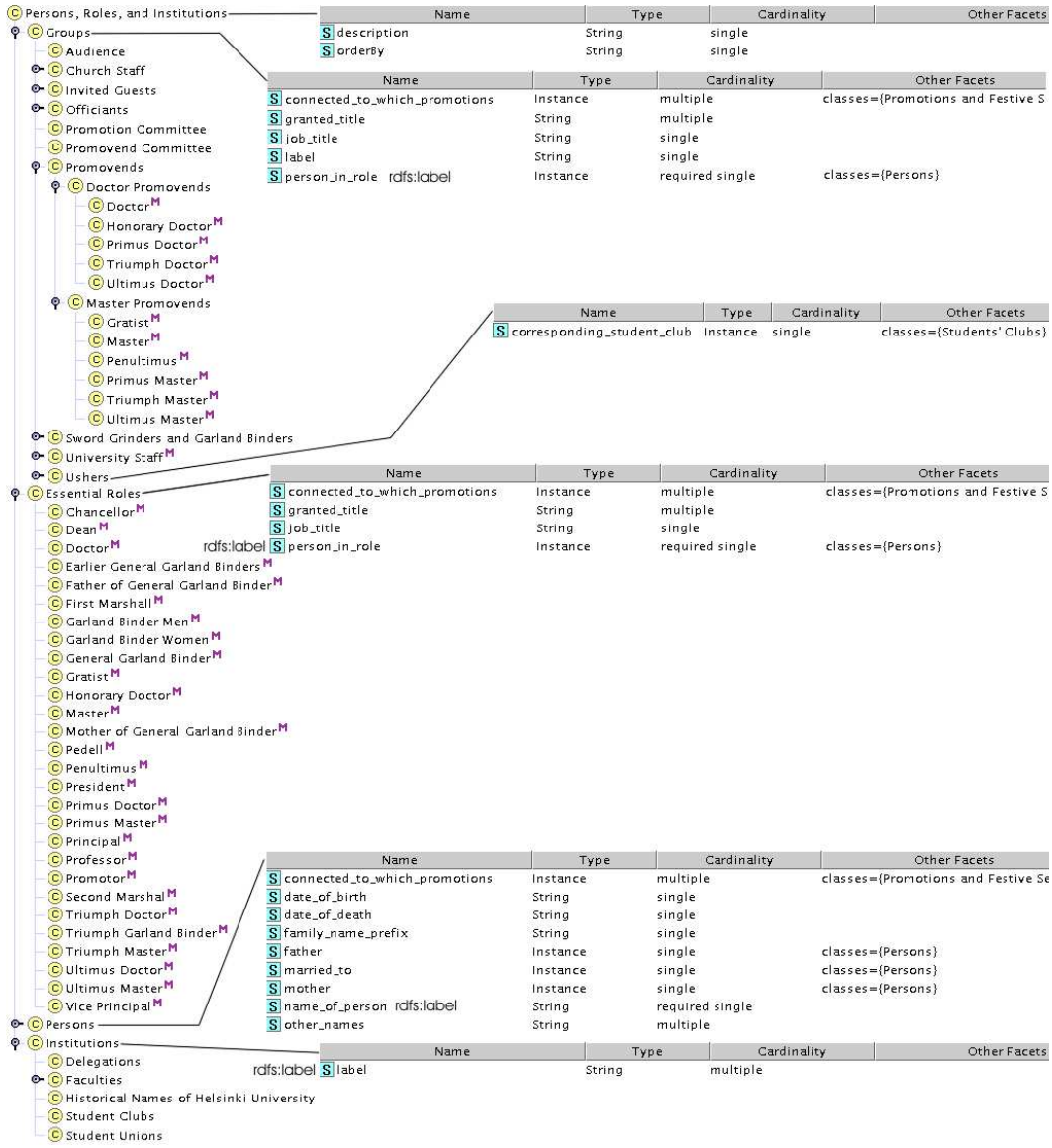


Figure 22: Structure of **Persons, Roles, and Institutions**. `M` on the right side of a class indicates that the class has more than one superclass.

the philosophical soundness of the class structure an extra class could have been added between **Persons** and the alphabet-classes, such as **Persons categorized by the first letter of the family name**, but that would not have been intentional and would have caused an extra mouse click every time a user wanted to find a certain person through class **Persons**. The division to alphabets could also have been programmed in the UI of the exhibition software, but the final result would have anyhow been more or less the same.

Values of property *name of person* serve as values of the `rdfs:label` of the instances. The values were given in form `family name 1st name 2nd name nth name`. There could have have been distinct properties *family name*, *first name*, *second name*, *mostly used first name*, *nickname*, etc., but then the programmers would have had more work in knowledge representation. They should have had to connect up to six or more name-fields to reveal one name. Some persons have a *family name prefix* like 'von', or 'af'. This was added separately to the name that was shown in the exhibition in order to have correct alphabetical ordering of the persons' names. Property *other names* was created because of maiden names and other possible changes of name. Properties *mother* and *father* were created to support semantic reasoning (sect 4.3.1); mother-father relationships are alone sufficient for creating an exact family tree assumed that all participants in the chain of mothers and fathers are known.

Essential Roles collects different roles of persons that appear in more or less every promotion. Many persons participate to more than one promotion during their life in different roles: *person in role* allows one person to have many roles, and many persons can have the same role during one or more promotions. The `rdfs:label` of instances of successors of **Essential Roles** was based on the `rdfs:label` of the value of *person in role*, which in the end was based on *name of person*. Property *connected to which promotions* facilitated the organizational efforts of the ontologist who could check to which promotions a certain person or a person-role had participated to. Properties *granted title* (e.g. counselor, Ph.d.) and *job title* (e.g. managing director, researcher) indicate the social status of a person during the promotion.

Groups collects subclasses of **Essential Roles** in a hierarchy. For example, **Promovends** is subcategorized into **Master Promovends** and **Doctor Promovends**, and their subclasses are also subclasses of **Essential Roles**. **Groups** collects also relatively rare roles that naturally are not subclasses of **Essential Roles**. **Groups** also describes photos where are people whose identity or a specific role is unknown. If there was a group of promovends in a photo with unknown identities and it would be unclear which kinds of promovends they were, then the photo would be linked directly to class **Promovends**. The same principle was used with all successors of **Groups**. If an instance was needed to describe this kind of group, property *label* would serve as the `rdfs:label`, and not the *person in role* which would be unknown.

Institutions collects **Delegations**, **Faculties**, **Student Clubs**, **Student Unions**, and **Historical Names of University of Helsinki** in a hierarchy. Successors of **Institutions** were used as values of some successors of **Promotions and Festive Sessions** and **Physical Objects**.

Happenings

Happenings collects happenings of different nature together. All successors of **Happenings** have exactly three properties as depicted in figure 23. There are two or more paths to every bottom-most successor of **Happenings in Sequence** that collects happenings in a sequential hierarchy. The other subclasses of **Happenings** classify successors of **Happenings in Sequence** based on the type of happenings: **Audial Performances**, **Dance Happenings**, **Dining Happenings**, and **Processions**. **Unofficial and Rare Happenings** is an exception because its successors are not successors of **Happenings in Sequence**.

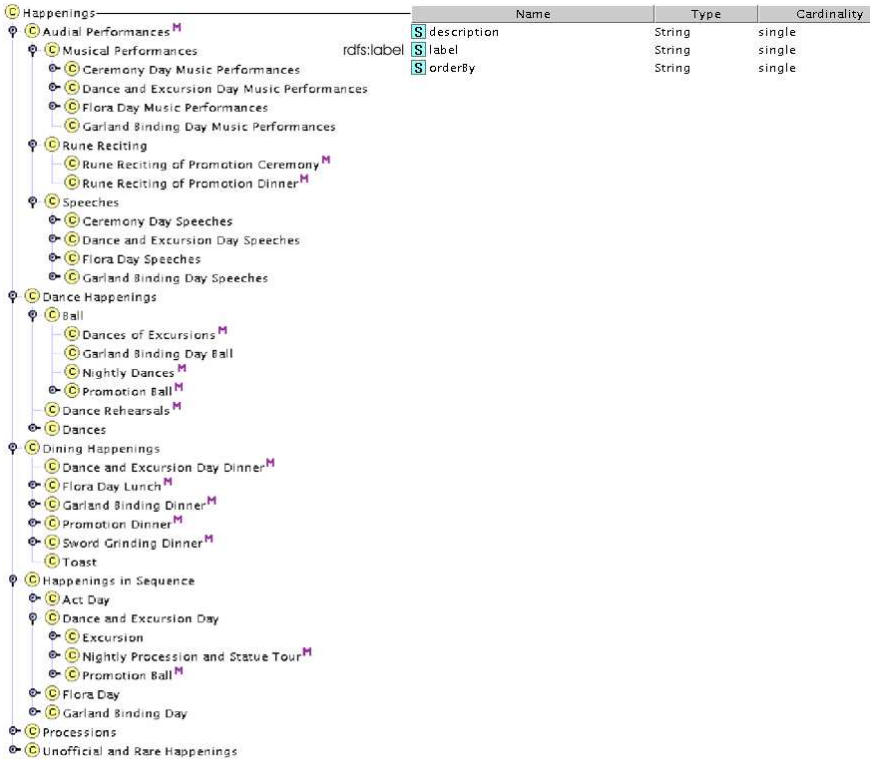


Figure 23: Structure of **Happenings**.

The happenings have had a relatively stable sequential ordering in promotions throughout the years: the four happening days are subclasses of **Happenings in Sequence**. **Act Day**, **Dance and Excursion Day**, **Flora Day**, and **Garland Binding Day** collect more condensed happenings that occur during those days, and so on. In addition to the hierarchy, the sequence of the happenings was specified by using the ASCII ordering [7] of the value of property *orderBy* that was defined for all classes and instances. When two categories **X** and **Y** have the same supercategory, and the happening described by **X** occurs before the one described by **Y**, the value of **X**'s *orderBy* is for example character '1', and **Y**'s character '2'. The ASCII order of '1' is lower than ASCII order of '2', and so **X** is revealed higher, or earlier to the user. This method is adequate to specify simple sequences but is not enough with more complex ones, such as with overlapping sequences. About two weeks of work of the ontologist and the programmers was wasted on experimenting too complex solutions to implement the sequential ordering, when the solution was eventually very simple.

Audial Performances is first subcategorized along the type of the performance. Every audial performance in promotions belongs to **Musical Performances**, **Rune Reciting**,

or **Speeches**. There are lots of different **Musical Performances** in all of the four main happening days, and so **Musical Performances** was subcategorized based on those. There are also lots of speeches in every happening day, so **Speeches** was subcategorized like the **Musical Performances**. There are however only two occasions where rune reciting usually takes place, and it would have been unnecessary to have categories for all four happening days. This is why **Rune Reciting** is subcategorized based on the name of the happening where the reciting takes place.

Dance Happenings was a bit unclear categorizing effort. It is subcategorized into **Ball**, **Dances**, and **Dance Rehearsals**. **Garland Binding Day Ball** was named after the day it takes place, but **Promotion Ball**, **Nightly Dances**, and **Dances of Excursion** all take place during **Dance and Excursion Day**. Creation of an extra class, **Dance and Excursion Day Ball**, would have anyhow been unnecessary because there are altogether only four ball-happenings, and no ball-happening occurs during **Act Day** or **Flora Day**. **Dances** was also a trouble spot. It describes the dances that are usually danced in promotions, but dances are not happenings. Placing **Dances** as a subclass of **Dance Happenings** was again a pay-off case where an easy optimal was reached by letting intentionality override philosophical soundness. It would not have been intentional to make **Dances** a top level class, and **Dances** would not suit as a subclass of any other top-level class. However, a user can find **Dances** by navigating to **Dance Happenings**, but the user might be misleaded because the first selection has to be **Happenings**.

Dining Happenings was subcategorized based on the official names of the dining happenings, or on the happening day if there was not an official name. An exception is **Toast**[26] that describes all the toasts in promotions. **Processions** is subcategorized based on the names of the places where the processions are heading to. **Unofficial and Rare Happenings** is subcategorized simply by the unofficial name or description of the unofficial happening.

Physical Objects

Physical Objects (fig. 24) appeared to be the most problematic top level class in sense of the philosophical soundness of the class structure, because the concept 'physical object' covers everything except abstract things (sect. 3.1.3). All **Physical Objects** have the basic properties *description*, *label*, and *orderBy*.

Assorted Objects collects objects with divergent nature. The only successor of **Assorted Objects** with extra properties is **Paintings and Photo Albums** that are *situated in* **Places** and have **Artists** as *manufacturers*. In some sense photo albums belong to **Printed Matter**, but since these photo albums were manually constructed in the early 20th century, they do not belong to **Printed Matter**. This indicates that **Printed Matter** collects only things that are automatically printed in a printing house, an information stated also by the class' `rdfs:description`.

Flags, Marks, and Badges collects all physical objects that identify institutions and show persons' rank, role, or any kind of granted or inherited social status. There are lots of traditional **Decorations**, **Flags**, **Seals and Heraldic Symbols**, and **Badges** visible in promotions. **Ushers' Ribbons** have from one to five colors, and **Ushers** belong to different **Student Clubs**. **Flowers** bring color to promotions, as well as

[26] A toast is a drink that is proposed to honor someone or something.

Figure 24: Structure of **Physical Objects**.

subclasses of **Headgear**, of which many are also subclasses of **Symbols of Academic Rank** and **Symbols of Office**. **Sculptures** have *manufacturers* and they are *situated in* different **Places**. Different kinds of **Vehicles** are used in promotions, such as **Car**s, **Ship**s,**Tram**s,**Buss**es, and **Horse Wagon**s.

Clothes and Dressing was first a top level class but it was forced as a subclass of **Physical Objects** due to its small size. **Clothes and Dressing** was changed several times during the process. First **Clothes** was subcategorized into **Co-ordinated Outfit** that stands for complete outfits like dinner jacket and tail coat, and into **Garment** that stands for single garments. **Garment** was again subcategorized into **UpperPart** and **LowerPart**, as recommended in ICOM standard [75]. The subclasses of **UpperPart** and **LowerPart** were categorized into general types based on which parts of the body they cover. Later that categorization proved to be too complex and the result was to subcategorize **Clothes** into **Headgear** that has many subclasses, and to set all the other clothes as direct subclasses of **Clothes**. **Dressing Instructions** are not **Physical Objects** or **Printed Matter** even though they can be printed on a paper, but a user can intuitively find **Dressing Instructions** through **Clothes and Dressing Instructions**. The problem is that it is unusual to search for dressing instructions starting from **Physical Objects**.

Performances, Performers, Creators, and Works

Performances, Performers, Creators, and Works (fig. 25) was constructed in course of providing an interesting view to the exhibition for people who are interested in performances and about things strongly related to performances. Photographs about different performances constituted also a large part of all the photos to be annotated. The class hierarchy consists mainly of successors of **Happenings** and **Groups**. The only subcategory of **Performances, Performers, Creators, and Works** that is not a successor of any other top level class is **Pieces of Work**.

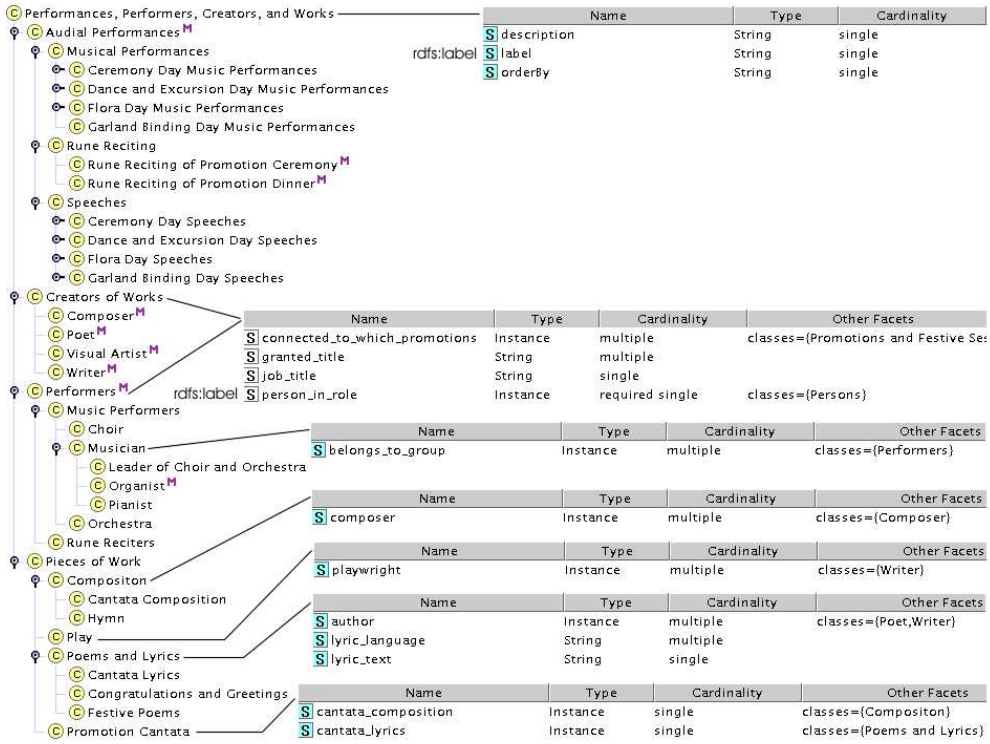


Figure 25: Structure of **Performances, Performers, Creators, and Works**.

Audial Performances are performed by **Performers**. **Performers** perform **Pieces of Work** that are created by **Creators of Works**. This way the structure describes different aspects of performances. All subclasses of **Creators of Works** are also subclasses of **Persons, Roles, and Institutions** > **Groups** > **Invited Guests** > **Artists**.

Conclusions of the Analysis

The whole process of ontologization and annotation took less than four months, but the structure of the annotation schema (sect. 4.2) was finalized when there was only a couple of weeks left for the ontologization and annotation. The annotation schema concretized the usage of the ontology and made some resolutions too complex and useless. This lead to removing many classes, properties, and instances. For example, in the early stages of the ontologization **Audial Performances** had property *performed by*, but it was discarded because the same information was obtained by linking an instance of successors of **Performers** directly to an image in the annotation stage.

The result was a simple RDF Schema that still had many properties that were not used with the exhibition (sect. 4.3). The top level classes of the promotion ontology fit

describing happenings of very divergent nature[27]. All happenings occur in some place(s). Traditional happenings naturally have a long tradition, i.e., they happen in a yearly basis, or follow some other period. Persons in different roles participate to happenings, and the happenings have subhappenings. There are clothes and other physical object present in happenings, and happenings have performances and performers who perform pieces of work. The similar kind of top level class structure with the properties can be used in ontologization of happenings in general, and also many subclasses of the top level classes suit for describing happenings in general. In a nutshell, the highest levels of the hierarchy are the most domain-independent, and the lower levels are more domain-dependent. This is very natural and the same kind of phenomenon can be seen with more or less all frame-based or category tree-based ontologies.

4.2 Ontology-Based Image Annotation

This section examines structure-based annotation with frame-based ontologies, continuing sections 2.3 and 3.1.5. Section 4.2.1 discusses how the annotation process needs to be guided and constrained by the annotation system in general level. Section 4.2.2 explains the schema that is used in annotating images with the promotion ontology, providing a case example of annotating one image.

The properties of frame-based ontologies (`rdf:Property`) are again called *fields* just as with the field-based and structure-based paradigms in section 2. Every annotation schema must have at least one field where the annotator can insert values. The values of the fields are selected from a category tree. Both classes and instances of frame-based ontologies are again called *categories*: the dichotomy is useless from the annotator's point of view since both classes and instances of RDFS can have same kinds of fields and same kind of values for the fields. A category tree that is used in selecting values for a field can denote a set of distributed ontologies, a set of branches of a single ontology, or any combination of these.

4.2.1 Constraining the Annotation

With the simplest structure-based annotation schema the annotator selects a top level category from a category tree and follows a promising path to find the most suitable category as a value of the field. When one category is selected the annotator starts again from the top until an adequate amount of categories have been selected. The navigation through the tree can be intuitively clear and provides the contextual information needed to retrieve the image later on. However, with complex schemas the annotation process can be hard. Annotation and ontologization cannot be cleanly separated because categories can possibly be created, deleted, related, and edited in many ways in both processes. The process starts by choosing one field from within a set of many fields. The annotator selects categories from a tree as values of the field or navigates through the tree and creates a wanted kind of category. The created category can always have a set of fields that can take categories as values: the categories that are selected or created as values can again take categories as values. In theory the cycle could go on forever even with an ontology with only one category to start with:

Set of fields → Category tree → Set of fields → Category tree ...

[27] Class **Promotions** should naturally be renamed after the happening that it describes.

Without any constraints the responsibility of coherent annotations lies totally on the annotator, which is why the annotation system needs to somehow guide the annotation process, constraining the categories and fields presented to the annotator. The deterministic methods of constraining the annotation are 1) cardinality constraints, 2) range constraints, 3) tree pruning constraints that take annotators' actions in account, and 4) editing constraints that specify annotators' rights to edit the category trees.

Cardinality Constraints

The cardinality constraint states the minimum and maximum amount of values that one field can have. The minimum and maximum amount of values can be stated with a closed range $[min\ max]$. [0 ∞] states that the field can have any amount of values. [2 ∞] states that the field must have at least two values, but there is no upper limit to the amount of values. [2 2] states that the field must have exactly two values, and [2 4] states that the field must have at least two values, can have three values, but cannot have more than four values. [2 4 6] states that a field can have two, four, or six values, and cannot have for example three or five values.

Range Constraints

The range constraint states what kinds of values a field can have. When the range is $\aleph$, the annotator can select any natural number as the value of a field. When the range is *ASCII*, the user can select *ASCII*-characters as the value of the field. The range can also be $ASCII > \{A, B, C..., Z\}$, that means that the annotator can select only upper-case ascii characters from A to Z.

Tree Pruning Constraints

The search space can be constrained based on the annotators' actions by using basic tree pruning methods. When the annotator has selected a set of categories as values of the fields, the paths to them can be closed by the annotation system so that the amount of categories that can be selected next decreases in every step. Some examples of pruning with the category trees on figure 26 are examined. Let the first selection be category **C**

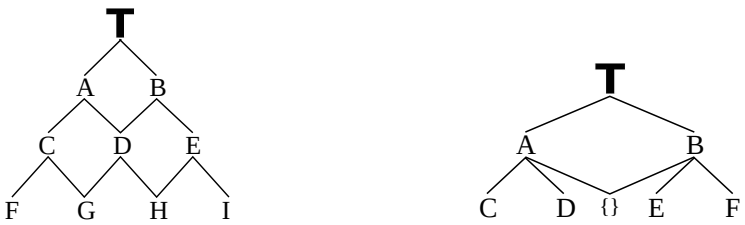


Figure 26: Two category trees used in annotation.

of the tree on the left. In the general case it is rational to specify that the annotator cannot select any of **C**'s predecessors or successors for describing the same thing. For example, if **C** was **Human**, then **F** and **G** would further classify the type of the human. If also **F** or **G** were selected it would be intentional to discard **C**. Selecting **A** would also be irrational if **C** was already selected: if **C** was **Human**, nothing would be gained by

annotating **A**, which could be for example **Mammal**. The rational choices after selecting **C** would be **B**, **D**, **E**, **H**, or **I** because they are not in successor-predecessor relationship with **C**.

With the strict definition (p. 17) two or more subcategories of category **X** share relatively nothing in common and their intersection is empty, as can be seen in the tree on the right of figure 26. These strict pruning rules could be used when describing positioning in a contest for example: normally when a person wins a gold medal the same person cannot win a silver medal in the same contest. If the annotator selects the gold medal as a value of a field that indicates a person's positioning in a certain contest, another medal cannot be selected as a value for the same field. The pruning rules can be applied in every level of a category tree in different ways. Some rule could constrain the annotation of a certain tree, a certain branch of a tree, or only a set of categories that are not in predecessor-successor relationship.

Editing Constraints

Constraints can be set on creating and using categories to avoid too long or cyclic paths in annotation, and to differentiate the annotation from ontologization. Different annotators can be given different rights to edit trees or parts of trees in a similar fashion than with traditional file systems. Let integer x denote the constraint. When $x = 0$ the annotator cannot create a category. When $x = 1$ the annotator can create a category ${C_1}$[28], but cannot create more categories as values of fields of C_1. When $x = 2$ the annotator can create a category C_1 and create another category C_2 as a value of a field of C_1, but cannot create categories for values of the fields of C_2. With $x = n$ the set of linked category-field pairs can have up to n members.

Constraints can be set also on using the categories that existed before the annotation started in a similar fashion, and the constraints can be combined. For example, in triple $[x, y, z]$ x denotes the constraint on creating categories, y on using the already existing categories, and z for maximum amount of using both existing and new categories in the linked chain of category-field pairs. The constraint $[1, 2, 2]$ specifies that the annotator can create one new category in the linked chain and can use also two already existing categories, but the maximum length of the chain is 2.

Combinations of Constraints

The discussed constraints can be combined to create unique annotation schemas. Let the goal be to create an annotation schema where the annotator is able to link n different types of specific metadata to an entity. One solution is to create a schema that contains n fields. Each field is mapped to a certain category, and when the annotator selects values for a certain field, a certain top level category is presented to the annotator. The annotator can, or has to select a certain amount of categories as values for each of the fields with certain pruning conditions. The annotation is completed when all the fields have been given an adequate amount of proper values.

[28]Note that C_1 can denote also a set of categories that are not in predecessor-successor relation.

Problems

Maintainability is the greatest challenge of structure-based annotation, assuming that there exists a commonly agreed ontology that can be used in annotation and retrieval. Possible modifications of the ontology that was used in the annotation can make the annotations totally or partly useless. When the ontology is modified, the annotation system and/or the administrator of the system needs to verify that the modifications are not in contradiction with the annotations that were created using an older version of the ontology.

As an example, let there be a category tree $\top >$ **Roundness**. The annotator has linked an image of a circle to category **Roundness**. When the tree is modified into $\top >$ **Roundness** > **Circle**, the annotation system should ask the annotator/administrator to verify whether or not the previously annotated image belongs under the new category. Then again, if **Circle** is deleted later on, the system can automatically link the image to the supercategory of **Circle**. With the categories only, the changes in ontologies can be handled quite easily assuming that there are enough annotators to verify the re-annotation of images. Handling changes of other relations of ontologies might then again require more complex verifications.

Annotators' errors are also naturally problematic as with all annotation paradigms. A lazy annotator can link an image about a circle to category **Roundness**, and when a retriever selects **Roundness** > **Circle**, the image is not retrieved. With systems that have thousands of annotated images these kinds of flaws are easily permanent, unless advanced feedback and computer vision techniques (appendix A) can be applied to verify placement of the annotated images somewhere in the future.

4.2.2 Annotation with the Created Ontology

The overall case was to annotate 628 photographs about the promotion ceremonies with the created ontology (sect. 4.1), and one photo is annotated here as an example. There were no constraints set on the annotation except that the top level classes could not be changed. This way the annotation can be seen also as ontologization, or as testing the ontology: many classes, instances, and properties were created, deleted, combined, and edited in many ways during the process. About 500 out of the 628 photos were previously annotated with field-based paradigm. These annotations were transformed into RDF format that could be used by the annotation editor[29], and the textual descriptions of the existing annotations were used as a reference in annotating the photos with the annotation schema in figure 27 in p. 50. The rest of the images were received from the UHe faculties with textual descriptions and scanned from books and from other sources.

The annotation schema enables linking every class and instance of the domain ontology to an image[30]. The schema was designed to describe different sorts of media formats, and so **Media Element** has properties that fit describing also video and audio in addition to photos. **Photograph** then again has special properties that suit for describing only photographs. As stated in section 1.2, some properties describe the abstract concepts that are represented by images' physical elements and some properties describe other

[29]Protégé-2000 was introduced in sect. 4.1.2.

[30]In fact, also the properties (`rdf:property`) could have been annotated, but these were not needed.

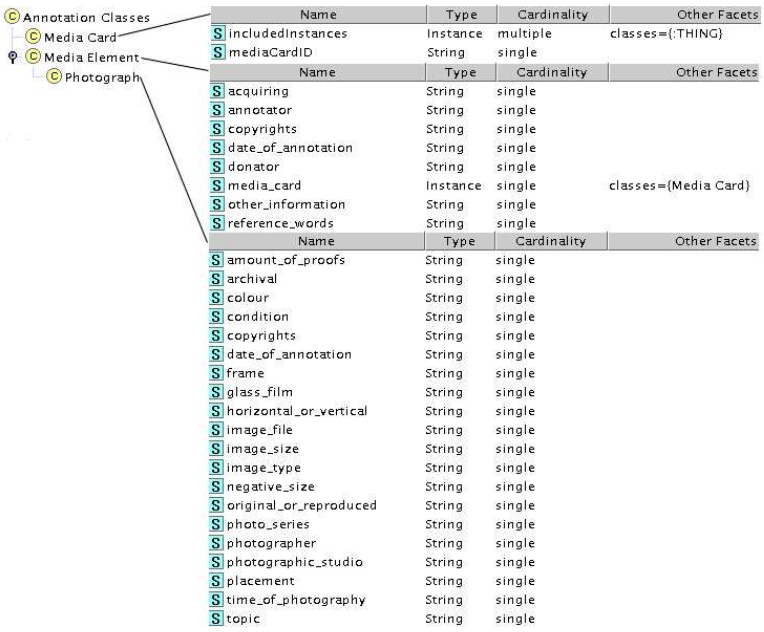


Figure 27: Classes and properties of the annotation schema.

things that are explicitly connected to the image. **Media Card** was used to describe the physical elements, in addition to property *topic* that was revealed in the exhibition but was not used in the retrieval. Also *reference words* describe the images' physical elements but neither this nor the other properties of **Photograph** were used in the exhibition, with the exception of *image file* that identifies the file names of the photos and *media card* that links an instance of **Photograph** to an instance of **Media Card**. **Media Card** has properties *mediaCardID* and *IncludedInstances* that links classes and instances of the domain ontology to an instance of **Media Card**. The `rdfs:range` of *IncludedInstances* is `THING`[31], and so also classes of the domain ontology could be given as values to *IncludedInstances*.

The only guideline in the annotation process was to describe the essential and interesting things in the images and to make the annotations as complete as possible, i.e., successors of every top level class should be annotated whenever intentional. The simplest solution to guide an annotator that has no earlier domain knowledge would be to create an annotation schema that has 6 fields that are mapped to the six top level classes of the domain ontology. However, the ontologist and the annotator was the same person there was no need create an annotation schema that guides the annotator. The subject of the annotation is the photo on figure 28 in p. 51.

The annotation starts by creating an instance of **Photograph** and giving textual values for the fields if these are not already set. Up to this point the annotation goes like with the field-based paradigm. After the needed fields have been given values the annotator creates an instance of **MediaCard** as value of *media card* and gives it an id such as Image12345.jpg.

[31]Protégé's `THING` corresponds to `rdfs:Resource` (p. 26) or $\top$ (p. 14).

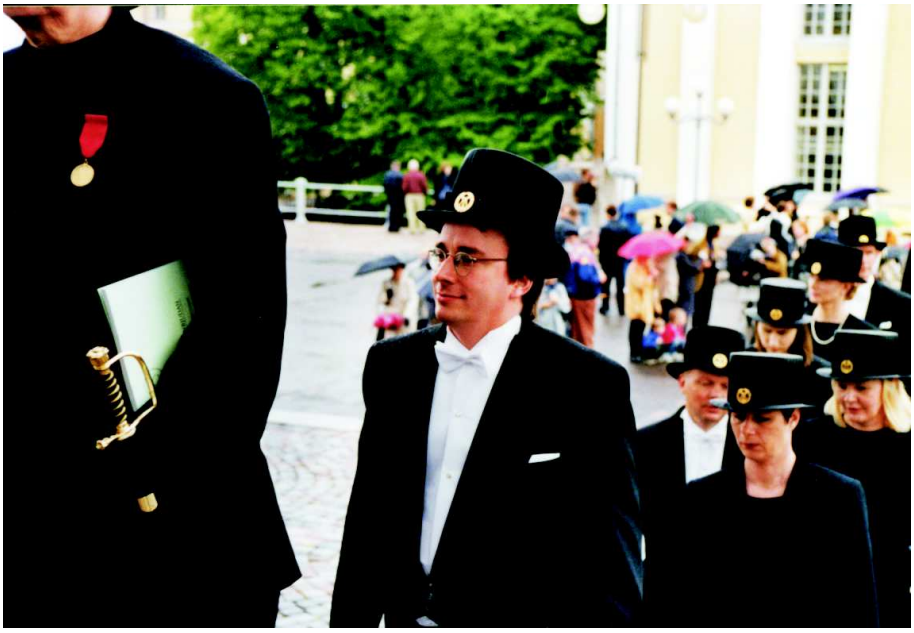


Figure 28: Image text: "Promotion of faculty of Philosophy. Honorary Doctor Linus Torvalds on a procession to divine service 2.6.2000."

The actual linking of the image to the domain ontology is done by selecting classes and instances of the domain ontology as values for *IncludedInstances*. The annotator has a set of top level classes to start with as depicted on the left side of figure 29. The annotator has to select one of the six top level classes and follow a promising path to find the most accurate class or instance. The annotator argues that the image is about a happening called 'procession to divine service' based on the image text of figure 28, and selects path **Happenings** > **Processions** > **Procession to Divine Service** as depicted on the right side of figure 29. If the wanted kind of successor of **Happenings** is

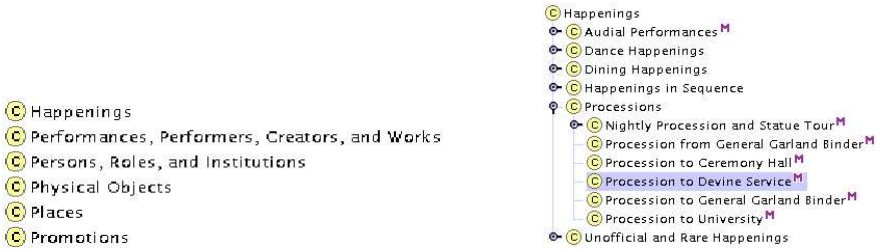


Figure 29: Top level classes on the left, and class structure of **Happenings** on the right. The path to the most accurate class is revealed, and the selected class or instance is highlighted with blue.

not found the annotator can create a new class or instance, or select a less accurate class or instance. In most of the cases, as with this one, more than one class and instance suit describing the subject image. After the first selection the annotator has to reason which top level class to browse next.

Every **Happening** occurs in a certain **Place**. There were many cases where it was hard to decide which places to select, as with the this one. There was only a small part of the National Library visible in the image but it was selected (fig. 30 in p. 52). because there were not any 'good' images taken about the library[32] and it was shown better than other places in the subject image. Only a small fraction of the ground level of the entrance

Figure 30: Annotation of **Places** on the left and annotation of **Persons, Roles, and Institutions** on the right. The integers indicate the amount of instances of the corresponding class. All instances of **Honorary Doctor** are not shown.

to Cathedral of Helsinki is revealed in the image. If the entrance was also selected the users would probably be misleaded: a user of a photograph exhibition assumes to see whatever she chooses to see and if a user chooses to see the entrance to Cathedral of Helsinki it should also be shown, preferably with a good image. Apparently there were lots of 'better' images about the entrance, which is also a reason why the entrance was not linked to the subject image. Also the Union Street (situated in between the library and the church) was not selected because it could not be seen at all in the image, even though the promovends are walking up to the church from that street.

The image text of figure 28 tells that **Honorary Doctor** Linus Torvalds is in the image. The annotator selects path **Persons, Roles, and Institutions** > **Essential Roles** > **Honorary Doctor** > Linus Torvalds, as depicted in figure 30. The same instance can be reached also via **Persons, Roles, and Institutions** > **Groups** > **Promovends** > **Doctor Promovends** > **Honorary Doctor** > Linus Torvalds. Selection of a successor of **Persons** would be unnecessary after selection of successor of **Essential Roles** or **Groups** because the information about the person in role is obtained from property *person in role* that belongs to all successors of **Essential Roles** and **Groups**.

If this was the first time that honorary doctor Linus Torvalds was selected the annotator would have to create an instance of **Honorary Doctor**, and give values for all the fields of the instance. Some of the fields (fig. 22, p. 40) take textual values but *connected to*

[32]This was know because the same annotator annotated all the images.

which promotions take successors of **Promotions and Festive Sessions** (fig. 21, p. 39) as values, and *person in role* takes successors of **Persons** as values. If the needed instances do not yet exist they have to be created. When all the needed fields have been given values up to an adequate degree, this part of the annotation is done, and in the next time the same person-role can be directly selected from the ontology.

Promotions and Festive Sessions provides the date when the conferring ceremony was held, the faculty that arranged the promotion, and much more information. Knowing the date 2.6.2000 it was trivial to select path **Promotions and Festive Sessions** > **Promotions by Century** > **Promotions in 21st Century** > 2.6.2000, as figure 31 depicts. If there were two or more conferring ceremonies in the same day the secondary search criteria would be the faculty because one faculty can have only one conferring ceremony in the same day.

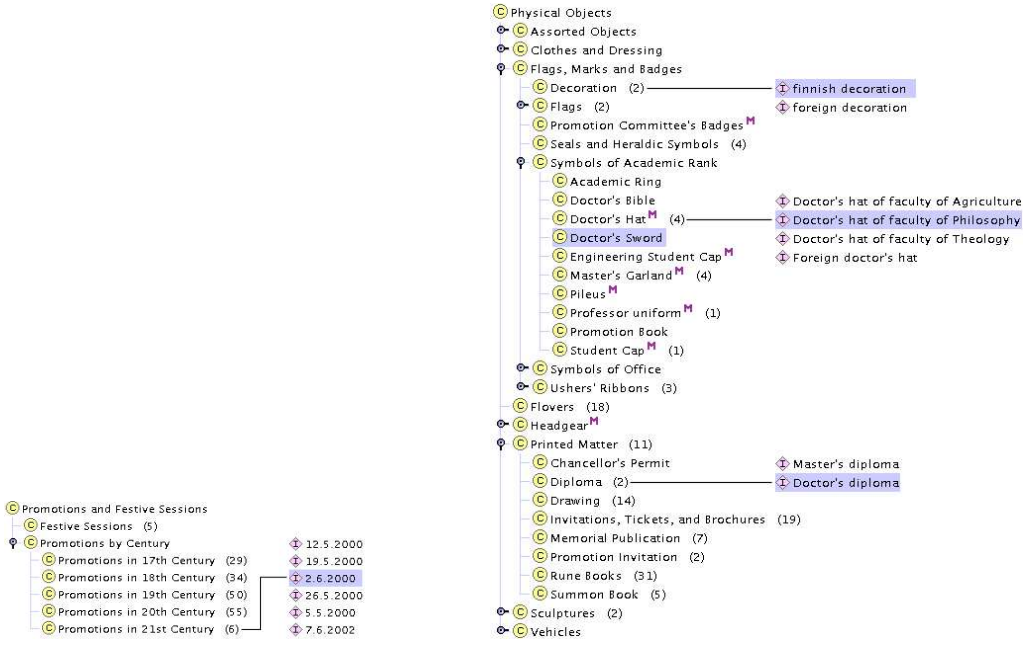


Figure 31: Annotation of **Promotions and Festive Sessions** on the left and annotation of **Physical Objects** on the right.

Annotating **Physical Objects** (fig. 31) was straightforward. The objects that can be seen in the image are selected. In this case there were many physical objects worth of selecting in the same image: the red decoration, doctor's sword, doctor's hat of faculty of Philosophy, and doctor's diploma. The knowledge that the diploma was doctor's and not master's came from the fact that the diploma was carried by a person with a sword, and only doctors have swords in promotions. Also the only sword in the ontology is **Doctor's Sword** that could have been noticed by an arbitrary annotator without any domain knowledge. If the annotator would not have any domain knowledge, had not seen the other images and image texts at all, and had no ability for semantic reasoning, the safest selections would have been **Decoration** and **Headgear**. The image retrieval system that uses the annotations is examined next.

4.3 Ontology-Based Image Retrieval System

This section examines and explains a fully implemented ontology-based photograph exhibition system called *Promotor*[33]. The system complements the ideology of structure-based (or view-based [151, 123, 16]) search by using the Promotion ontology (sect. 4.1) as an information retrieval structure and as the basis of recommending images. It was not known exactly what was the exhibition going to be like in the ontologization stage and some features of the ontology were not used at all in the exhibition. Even though formal ontologies can be used in many ways, the principles presented here are widely applicable with retrieval systems that use frame-based ontologies.

Figure 32 depicts the part of the application's interface that is revealed when the system is started. On the center is the current main image. Directly above the main image are four selection tabs: `Image` (currently active), `Search` (explained later), `Help` (user's manual), and `About` (info about the software). Below the main image are `rdfs:labels`

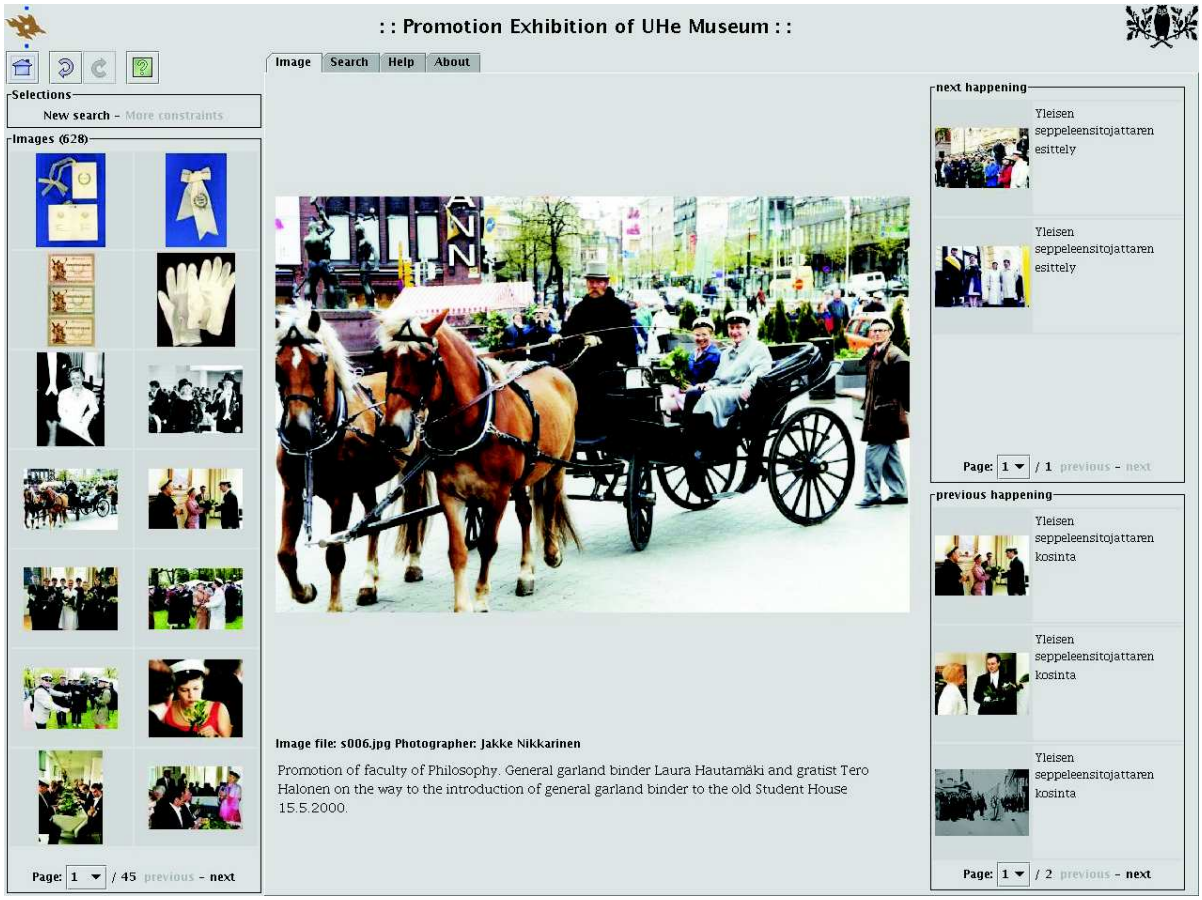


Figure 32: Image-page is revealed at the start.

and values of properties *image file* and *photographer*, and value of *topic*, i.e., the image text (fig. 27, p. 50). On the left side of the main image is a set of thumbnail images. The system contains thumbnails of each of the 'big' images. The thumbnails function as links, and by clicking a certain thumbnail-image it is magnified to be the main image on the center. The thumbnails on the left side appear in a random order, given the current

[33] *Promoottori* in Finnish. The software is currently installed into a kiosk-box, situated in the vestibule of the Museum of University of Helsinki, Snellmaninkatu 3. The software is examined also in [72, 73].

search constraints. When the system is started and no constraints are set the user can quickly browse through all the thumbnails in the system by clicking `previous` or `next` on the bottom left corner. The buttons above the left side thumbnails are familiar to a user of an Internet browser: `Home` resets the system in the default starting state, `back arrow` takes the system in the previous state, `forward arrow` takes the system into the state where it was before pressing the `back arrow`, and `?`-button reveals the help-page with identical functionality to the help-tab above the main image.

4.3.1 Using Inference in Recommending Photos

On the right side of the main image are the *recommendation panes* that reveal thumbnails of the images that are linked to the main image via different sorts of semantic relations[34]. Two panes out of five are currently revealed. All the panes have been given hard-coded priority values and if the priority level of a certain pane is not reached it is not revealed. The priority level is calculated based on the categories that were linked to the main image in the annotation stage.

Successors of every top level category of the domain ontology except **Performances, Performers, Creators, and Works** (fig. 25, p. 45) were linked to the current main image but the priorities of only two panes, `previous happening` and `next happening` were met, which is why they are the only ones revealed. The upper one of the current panes reveals thumbnails of the images connected to the next happening and the lower pane reveals images connected to the previous happening. The sequences of happenings are based on the value of property *orderBy* that was given for all successors of **Happenings in Sequence** (fig. 23, p. 42). Figure 33 depicts these values as categories. Let the current main image be linked to a successor of **Happenings in Sequence** that

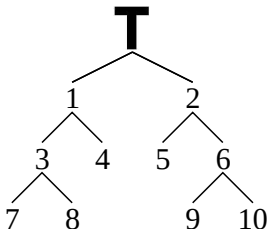


Figure 33: Values of *orderBy* as categories.

is denoted by 4. The `previous happening` pane would in this case reveal thumbnail images that are linked to 3 or to 3's successors 7 and 8. The `next happening` pane would reveal thumbnails that are linked to 5 that has no successors. If the current main image was linked to 7, 3, or 1, there would not be the `previous happening pane`, and if the main image was linked to 2, 6, or 10, there would not be the `next happening pane`.

The `person relations` pane recommends thumbnails based on the structure of **Persons** and **Groups** (fig. 22, p. 40). **Persons** has properties *mother* and *father* that take successors of **Persons** as values. Let **Persons** have two successors A and B, and let A be directly linked to the main image. If A's property *mother* has value B, then the images that are directly linked to B are revealed as thumbnails. Actually the properties

[34]Prolog [125] was used as an inference language to reason about the recommendations, i.e., the recommendation rules were programmed with Prolog.

that were used in reasoning about the person-relations were not explicitly specified. The only rule was to use the kinds of properties of **Persons** and **Groups** that take successors of **Persons** or **Groups** as values. This is an example where only the `rdfs:range` and `rdfs:domain` of the properties count, and the `rdfs:label` or the exact ID (p. 26) of the properties is totally disregarded. This reminds the reader of the fact that the meaning of an entity depends on relationships to other entities. The only pay-off with rules like this is the need to take the rules in account when editing the domain ontology[35]. When a thumbnail is recommended, the `rdfs:label` of the property that the recommendation is based on is also shown, like *mother* and *father*.

`Place relations` are based on the structure of **Places** (fig. 19, p. 36). If the main image is linked to an instance of a successor class of **Places**, then other images that are linked to the same instance are recommended.

`Promotion relations` are based on the structure of **Promotions** (fig. 4.1.8, p. 39). Thumbnails that are linked to the same promotion as the main image are recommended.

4.3.2 Structure-Based Search

The Search-tab (fig. 34) reveals six facets that correspond to the six top level classes of the domain ontology, while the left side of the interface remains the same.

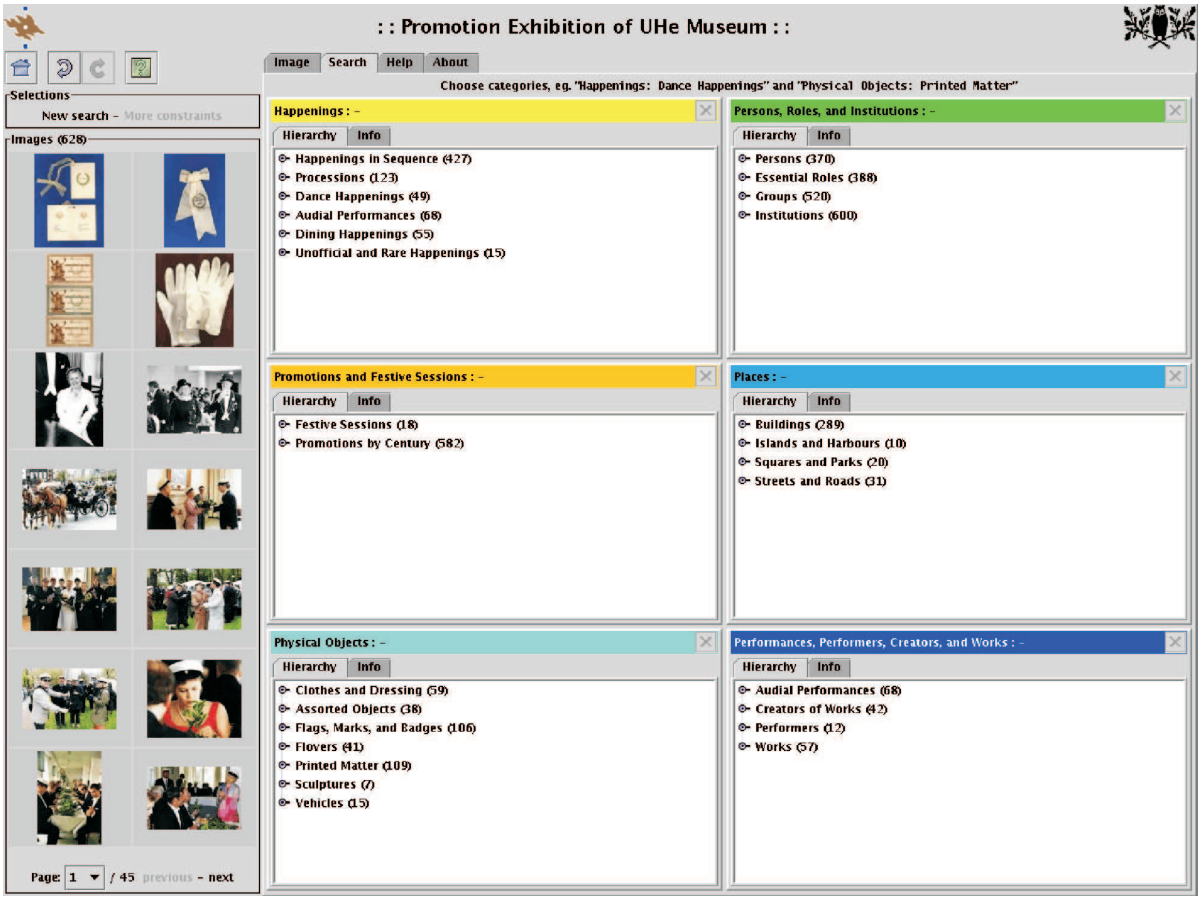


Figure 34: Representation of categories.

[35] A theoretical problem occurs when a property like *has got nothing in common with* is added.

The user can find the search menu also by pressing `New search` above the left thumbnail images, that resets all the facets into the initial state where no search constraints are set. The facets can also be reseted one by one by pressing the 'X' in the upper right corner of each facet. The `rdfs:label`'s of the top level classes are the titles of the facets. Subclasses of the top level classes are also revealed, which gives the user a direct visualization of two highest levels of the class hierarchy.

The dichotomy to classes and instances proved to be useless from the user's point of view, and therefore both classes and instances were given an equal representation in the UI, and are referred again as categories from here on. If there were no images linked to a certain category or to its successors the category was not revealed at all. As an example, category **Penultimus** is not revealed in figure 4.3.2. It belongs to the ontology (fig. 22, p. 40) but no images were linked to the category. The revealing and hiding of the category structure is done by clicking the node on the left side of a category.

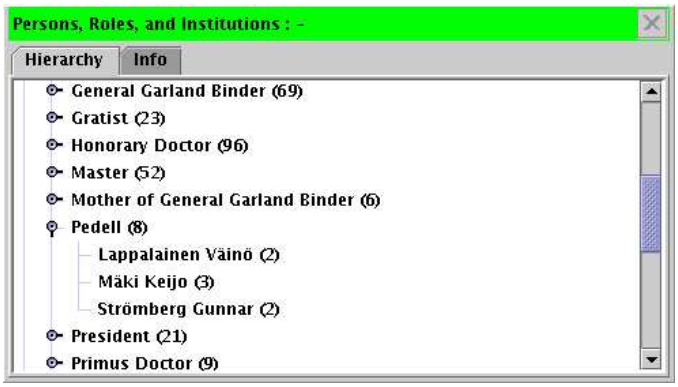


Figure 35: Integers indicate the amount of images under a category.

All categories *inherit* the annotations of their successors, i.e., a union is created of annotations of a category and annotations of its successors, as explained in section 3.1.5. An integer is attached to every category that indicates the total amount of images that are linked to a category and to its successors. For example, when category **Pedell** (fig. 4.3.2) is linked to one image the indicating integer of **Pedell** and all of its predecessors is increased by 1. There were 7 images linked to successors of **Pedell** and 1 linked directly to **Pedell**, which makes the indicating integer of **Pedell** to be $7 + 1 = 8$. Next, four examples of structure-based search are given, and compared to text-based search.

Search Example I

The goal in the first example of structure-based search is to retrieve images where can be seen honorary doctor Linus Torvalds. This can be done in the same fashion as the corresponding annotation (fig. 30, p. 52). The user chooses the facet with title **Persons, Roles, and Institutions** (fig. 34) and selects **Essential Roles** > **Honorary Doctor** > **Linus Torvalds**. A set of 5 photos where can be seen honorary doctor Linus Torvalds is retrieved (fig. 36). Photos about honorary doctor Linus Torvalds could be retrieved also by selecting **Groups** > **Promovends** > **Doctor Promovends** > **Honorary Doctor** > **Linus Torvalds**. The third path **Persons** > **T** > **Linus Torvalds** retrieves photos about Linus Torvalds disregarding his role[36].

[36]The photos would anyhow be the same because Linus was in role of honorary doctor in every photo about him that are annotated in the system, which is not the case with many of the other persons.

Figure 36: Search results of search example I.

Search Example II

The goal in the second example of structure-based search is to retrieve all those images where can be seen both general garland binder (**GGB**) *and* garland at the same time[37]. When a user selects a certain category **X** the search space is constrained so that only those categories and their predecessors are allowed to be chosen that are linked to the images that are members of $\mathbf{X}_a$[38]. This is visualized in the UI by changing the color of the categories that cannot be chosen into gray. In terms of set theory, the gray categories contain only annotations that are members of set $G = \top_a \backslash \mathbf{X}_a$, where $\top_a$ denotes the set of all images annotated in the system. This way the user does not have to browse those branches of the category tree where nothing can be found, given the constraints the user has already set.

Figure 37 depicts the retrieval situation in four steps. The user sees a set of categories

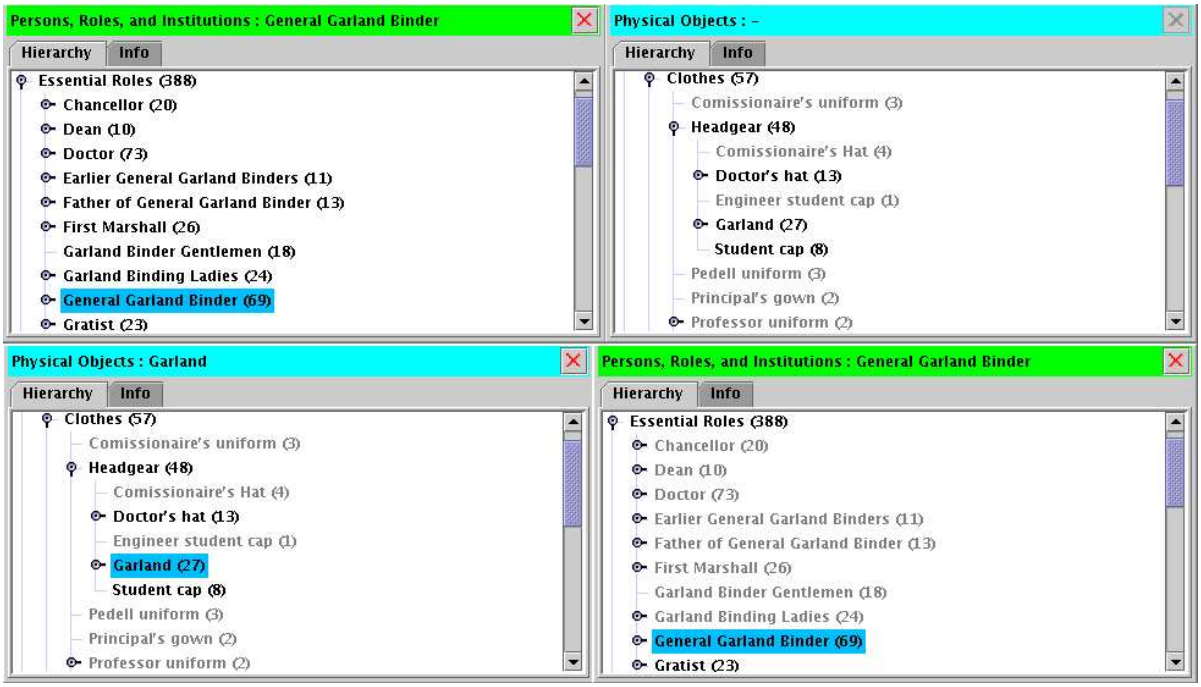


Figure 37: Four steps of the retrieval process: 1) top left, 2) top right, 3) bottom left, and 4) bottom right.

and selects **GGB** in the top left corner. Now all the images that are members of set $\mathbf{GGB}_a$ are revealed as thumbnails disregarding their membership to $\mathbf{Garland}_a$. The

[37]The goal was identical with text-based paradigm in section 2.1.2.

[38]As in section 3.1.5, $\mathbf{X}_a$ stands for the set of all images that were linked to category **X** or to **X**'s predecessors in the annotation.

selection affects other categories, and the **Physical Objects** facet on the top right corner is constrained so that the categories that are fuzzified with gray color cannot be selected. The user selects **Garland** in the bottom left corner, and the selection is reflected in the bottom right corner. Thumbnails of the 11 images where can be seen both garland *and* general garland binder are revealed (figure 38), and all of them are members of set $\mathbf{GGB}_a \cap \mathbf{Garland}_a$.

Figure 38: Search results of search example II.

Search Example III

The recall and precision (sect. 2.1.2) of structure-based versus text-based search is analyzed in the following. The structure-based search is executed with the exhibition software and the text-based search with Protégé-2000 (sect. 4.1.2). The text-based queries are compared with the *topic*-field (fig. 27, p. 50) that textually describes the images. The text-based search was executed using Finnish search terms[39] but the results would have been almost the same even if the texts were translated into English. The integers indicate the amount of retrieved images on the below table. With text-based search `AND` in the table corresponds to logical $\wedge$, and with structure-based search it corresponds to $\cap$. `OR` corresponds to logical $\vee$ and $\cup$ respectively.

Query terms	**Structure-based**	**Text-based**
`Garland`	27	124
`Garland binding material`	5	0
`Garland binding happening`	23	23
`GGB`	69	42
`GGB AND Garland`	11	42
`GGB OR Garland`	85	124

In the first query `Garland` the precision of text-based search was very poor. The irrelevant images within the 124 images are those where cannot be seen garland but they are about

[39]The categories that were used in the comparison and the corresponding search terms in Finnish: **Physical Objects** > **Flags, Marks, and Badges**, **Symbols of Academic Rank** > **Garland** = Seppele. **Persons, Roles, and Institutions** > **Essential Roles** > **General Garland Binder** = Yleinen seppeleensitojatar. **Physical Objects** > **Assorted Objects** > **Garland Binding Material** = seppeleensitomismateriaali. **Happenings** > **Happenings in Sequence** > **Garland Binding Day** > **Garland Binding Happening** = seppeleensitojaiset.

1) election of GGB, 2) parents of GGB, 3) previous general garland binders, 4) triumph garland binders, 5) the companions of the master promovends who are called garland binding men and garland binding women, and 6) garland binding day when there is mainly only garland binding material visible. The recall of the text-based search was good, but finding the wanted ones by browsing through 124 images would take relatively much more time than with structure-based search.

If the retriever's goal was to see all kinds of garlands, including the partly unfinished ones, the recall of text-based search would have been even better than with structure-based search. This is because there is an additional category **Garland Binding Material** in the ontology for describing unfinished garlands and the material that is used to bind the garlands. If the user had unnoticed this category the unfinished garlands might have been totally unseen. Then again, with text-based search the query `Garland binding material` would not retrieve images at all. To find the unfinished garlands with text-based search the user should know that they can be seen during the garland binding happening. The query `Garland binding happening` retrieved 23 photos with both paradigms, but only the 5 clearest photos about the garland binding material within the 23 photos could be retrieved with structure-based search by selecting **Garland Binding Material**. However, text-based query `Leaf OR Branch OR Laurel` retrieved 17 images of which half were totally irrelevant but some contained garlands that could not be retrieved with structure-based search. The reason for this is that those images that were linked to **Garland**, to **Garland binding material**, or to their successors depict the garlands and materials more clearly than the other annotated images.

The query `General garland binder` retrieved more images with structure-based search than with text-based search. The precision was perfect with both paradigms but the recall of text-based search was worse. This is because in Finnish the inflectional stemming of 'general garland binder' changes quite much when used with expressions such as `GGB's`, `to GGB`, `from GGB`, etc. The retrievers should use different word forms with text-based search to achieve the same recall, but then the precision could get lower.

With text-based search the query `GGB AND Garland` retrieved exactly the same 42 images than with query `GGB`, which naturally indicates a low precision. This is because the character string "garland" is already included in "General garland binder": this is a classical situation where the advantages of structure-based search are clear. The recall was almost identical with both paradigms, but the precision of structure-based search was again 100%. However, there were 2 images in the system where the GGB can be seen with an unfinished garland in her hand, and these images do not belong to the set of 11 images that were retrieved with structure-based search. These two images could however be retrieved with structure-based search by selecting `GGB AND Garland binding material`.

The query `GGB OR Garland` retrieved the same 124 images with text-based search than with query `Garland`. The query `GGB OR Garland` could not be directly executed with the exhibition software, and so the union $\mathbf{GGB}_a \cup \mathbf{Garland}_a$ was executed manually by two separate structure-based queries: `GGB` that retrieved 69 images and `Garland` that retrieved 27 images. Some of the retrieved images were the same with both queries, and the total amount of different images was 85.

Search Example IV

Another example of comparing structure-based and text-based search is given in course of objectiveness. This time the retriever's goal is to find images about processions of different faculties as depicted on the below table[40]:

Query terms	**Structure-based**	**Text-based**
`Procession AND Political Science (valtio)`	10	10
`Procession AND Law (oikeus)`	3	3
`Procession AND Veterinary Medicine (eläin)`	1	1
`Procession AND Medicine (lääke)`	9	10
`Procession AND Agriculture (maatalous)`	3	4
`Procession AND Philosophy (filosofi)`	95	56

The first three queries retrieved exactly the same images with both text-based and structure-based search. The recall and precision were also 100% with both paradigms because the search terms did not contain one another and the search terms were not used together in other contexts. The text-based query `Procession AND Medicine` however retrieved also one image about procession of faculty of Veterinary Medicine because character string "Medicine" is included in "Veterinary Medicine". The structure-based query `Procession AND Agriculture` retrieved one image less than the text-based query because the annotator had not linked **Faculty of Agriculture** to the image; a flaw that was noticed during this test. The text-based query `Procession AND Philosophy` then again retrieved much less images than the corresponding structure-based search. This is because the query terms do not appear in all the description fields of images that are about processions of the faculty of Philosophy. The results are further analyzed in the next section.

[40]The categories and corresponding search terms in Finnish: **Happenings** > **Processions** = kulkue. The faculties can be found in the end of path **Persons, Roles, and Institutions** > **Institutions** > **Faculties** > **Promotion Faculties** >*. Finnish search terms for the faculties are inside the brackets in the table.

5 Conclusions

This section analyzes the fitness of ontology-based approach to image annotation and retrieval in scope of the case example and in domains of the similar size. The ontology-based approach is compared to the text-based approach in building a domain domain model, annotating and retrieving images, and reusing the annotations. The extensibility of the ontology-based and text-based approaches are analyzed in scope of the Internet.

Building a Domain Model

In light of the case example, formal frame-based ontology languages with a variety of supporting tools provide an efficient way to build and use a domain model. Disregarding if the domain is an ontology or a thesaurus, the new framework of Semantic Web languages and tools are more efficient in the task than traditional programming languages; naturally, a domain model cannot be built with natural language, but a domain can be described with natural language. Nowadays there are not the kinds of globally agreed ontologies that could be easily used as the basis of creating domain-specific ontologies such as the Promotion ontology, and so the ontologization has to be started from a scratch, but as the case example proves on it's own behalf, this is not a great problem and can be done efficiently with the aid of domain experts and the ontologist.

Annotation

With traditional text-based annotation the annotator has to have an explicit thesaurus to ensure that the right words are used. If the relations of the words in the thesaurus are formalized, the thesaurus would be moving towards a formal ontology. With structure-based annotation the ontology *is* the thesaurus, and ontologies facilitate guiding the annotators. Given a domain ontology, an annotation schema can be easily built with which the concepts that are used in the annotation can be constrained so that the annotator's job is on better grounds. Complexity of the annotation schema can increase along the complexity of the ontology, but the annotation schema can still be intuitively used by annotators with no domain knowledge. For instance, it was very easy to build and use the annotation schema of the case example, and if the annotator and the ontologist would not have been the same person, it would have been trivial to create a schema where are six separate for each of the six top level classes of the Promotion ontology.

When new data is included into the ontology during the annotation (e.g. persons and places) it can be easily reused, when in contrast text-based annotations have to be created manually from the beginning until the end, no matter how many annotated items there are. Text-based annotation can however be applied to any domain simply by inserting the annotation text into one annotation field, when in contrast ontology-based annotation requires an ontology that is designed for the subject domain. Also, an ontology-based annotation schema has to be created according to the used ontology (or ontologies), and if the design principles of the ontology are modified, the annotation schema has to be modified accordingly: handling of the changes in ontologies is a great task that is yet undone.

Retrieval

The Promotor retrieval system provides better means than text-based search to formulate the information need, formulate the query, and to understand the result set. With structure-based search the actual retrieval includes only finding the search constraints, categories that really describe what is in the images, after which the search is done and the user can concentrate on the actual image data, which naturally is the goal of an average user of a photograph exhibition system. The inheritance of annotations from the bottom of the hierarchy up to the top of it, and the thumbnail images assure a rapid functionality. When the search results are depicted on the same window than the categories the iterative query formulation gets easier. The feedback is instant and users understand the size and the contents of the search space.

In general, the set of 628 images is so small that the recall is close to 100% with both text-based and structure-based search with many queries, assuming that the retriever is familiar with the domain. The precision is usually better with structure-based search but it naturally depends on how users can find the search constraints through the category hierarchy. With a small subject domain the hierarchy is clear and the recommendation system also guides the retrieval. If the system used only text-based search, ceteris paribus, ignorant users would first have to learn about the domain by reading the image-texts or other descriptions, and then type the queries accordingly. Typing a short query like `Garland` takes less time than finding the corresponding category, but an imprecise query formulation leads to low precision. With low precision the retriever has to use time in browsing irrelevant results, or formulate a more precise query iteratively by viewing the images and image texts of the result set. It is certainly easier, faster, and more entertaining for an ignorant user to start from high-level abstractions by selecting categories with a mouse from a set of possible choices and following interesting links that really lead to the wanted data.

The user has possibly no exact goal in mind, when the category hierarchy and the semantic recommendations can also teach the user about the domain more effectively than text-based search. Again, a textual thesaurus could be revealed to the user, but in this case it would be intentional to allow the user to choose the search constraints directly from the thesaurus, which was done with the ontology in the case example. Field-based approach could function more efficiently than generic text-based search, but then the annotations would have to be created accordingly. Implementation of the recommendation system would also get a lot harder, and as the classical problems of text-based search prove, the precision of the recommendations would be far from perfect: explicit thesauruses or manually created hyperlinks would have to be used, when in contrast the RDFS ontology with Prolog rules served successfully as the thesaurus and the link-structure in the case example.

The reason to the outranking precision of structure-based search and the recommendation system is the fact that the actual annotation process captures the annotators' intentions implicitly, which is not the case when text-based retrieval is applied to text-based annotations. When the same ontology that is used in the annotation is used in the retrieval, intersection, union, and difference can be deterministically applied to sets of categories, directly or via different relations. These were used in two ways in the case example, automatically as embedded functionality, and as retriever's selections. As embedded functionality, union was used in inheritance of the annotations, and difference and intersection in visualizing and constraining the search space. The retriever could

constrain the search space only with intersection, and it is notable that difference and union could have been used also as retriever's selections in the structure-based search. A retriever could benefit a lot by selecting for example all garlands, but not those garlands that appear during the garland binding day. Union is then again less useful as retriever's selection than intersection and difference: the same items that can be retrieved by selecting the union of two categories can be retrieved by selecting one category at a time.

Formal ontologies facilitate using different types of data directly as search constraints. In the case example thumbnail-images were used as hyperlinks, but also videos and sound files could have been used in a similar fashion: a retriever can choose between different types of music based on very short sound samples. Representation of information as it really is provides logically easy means for humans to retrieve it. As a conclusion, ontology-based approach seems to be applicable to relatively small and closed domains in image annotation and retrieval better than text-based or field-based approaches, and building a domain model is more effective with standard frame-based ontology languages and supporting tools than with traditional programming languages.

When the Domain is the Internet

The size of the domain crucially affects the design principles of ontologies and annotation and retrieval systems. When the domain gets wider, it can be assumed that the same bottlenecks and advantages can be distinguished in a larger scale with both text-based and structure-based paradigms. When the domain is the whole Internet, structure-based and text-based methods have to be strongly combined to reach better results than earlier. In the following, traditional text-based retrieval systems are compared to today's largest structure-based on-line annotation and retrieval system Open Directory Project ODP[41].

Extensive studies [99, 35] show that users who know exactly what they want can reach good results with retrieval systems that have only one text-based search field. To date, Internet search engines such as Google [55] and Altavista [2] are a lot more effective than ODP in finding information in general. This is not a surprise: Google alone has over 4 billion indexed Web pages and ODP has only about 4.5 million Web sites, even though there can be many separate pages on one Web site. However, as the size of ODP grows, also the usage can grow simultaneously. The precision of ODP is a lot better than that of Google because of the same reasons than in the case example: the precision of structure-based search that is based on manual annotations does not get lower as the amount of annotated items grows. The structure-based search is more on the categories and not on the content, because once the categories have been found, the intended content is also found. With small domains the categories themselves can be found in a normal structure-based fashion, but with large domains text-based search helps to find the categories by their titles and descriptions: a textual query is typed, after which a set of categories is represented to the user. The titles of the categories that are represented to the user contain the query string, and/or the sites that are placed under the categories that are represented to the user contain the query string. By selecting a category the search space is effectively reduced based on the predecessor-successor relationships of the selected category. With ODP one can search either only categories, or sites that are constrained by the selected category.

[41] ODP is explained briefly in appendix C. The directories of ODP can be taken as a category tree that is used in annotation and retrieval of Web sites.

Ontologies like ODP can be used to link Web sites to general domains, but it seems to be rather a heavy task for an average site owner to annotate all images and pages separately with today's tools. This indicates that text-based methods accommodated with semantic relevance feedback [105] that is based on retrievers' average actions can be applied more successfully to image retrieval in the near future than to use explicit ontologies in the annotation. If only ontologies were used without any relevance feedback, the best precision in image retrieval would be reached by annotating the Web sites where the images are, and the images separately. If the images were not annotated separately, relevance feedback would have to be used again. However, creation of the relevance feedback system would be on far better grounds even if only the Web sites were annotated. One clear advantage of the text-based paradigm is the fact that there is not a need to build explicit domain ontologies, that is a prerequisite for manual structure-based annotation and retrieval.

With the text-based paradigm there is not necessarily a need to go through an explicit annotation process. One can write text on a Web page and put an image there that can be possibly found with text-based search. Text can be inserted also into a metadata field of a Web page or an image. Plain-text annotations can be reused simply by copying the text to another place, but structure-based annotations can be reused only with the ontology that was used in the annotation, and with a system that is able to interpret the ontology. Then again, if the ontology had been built using globally agreed categories and languages and the ontologies could be placed in the Internet 'permanently', that is the goal, the annotations would always be globally valid.

An ontology that covers the whole Internet should relate all words of a natural language like English, and ontologization of a language in a way that it can be used efficiently in annotation and retrieval is a huge, if not an impossible task. Efforts in the past indicate that a closed community cannot build an ontology that covers the whole World effectively. An open ontology such as ODP then again benefits from volunteered editors, and it seems that in general people are willing to participate in ontologization efforts. However, the editors' subjectivity can also cause problems. As ODP-like ontologies evolve, it is highly probable that hundreds of millions of Web users do annotate their sites in order to get them found by a large audience, and creation of recommendation systems is on better grounds when there is a reliable category hierarchy to work with.

When the domain grows, also the pay-offs in the category structure get worse. ODP and the case example prove that it is better to use multi-instantiation, i.e., set the annotated items below more than one category than to use only multiple inheritance of categories and set the annotated items below only one category. This enables using intersection, union, and difference effectively to constrain the search space, and the ontologization is easier when most of the categories do not have to be enforced into a predecessor-successor relationship. When the amount of independent top level categories grows, the amount of unsound philosophical resolutions diminishes, and even if there were thousands of top-level categories, these too could be placed into a well-designed hierarchy. For example, there can be thousands of domain ontologies such as the Promotion ontology, that can all be placed under other higher-level categories. Naturally, changing for example ODP into a multi-faceted search system, such as the Promotor exhibition system is, would require fundamental changes in ODP.

Formal ontologies have come to stay permanently with us as a highly effective language of abstraction with which humans and computers can share information across domain boundaries. The benefits of ontologies are clear, and even if there would exist a 'perfect' linguistic system that could automatically annotate data based on long textual descriptions, it would not solve the need for structure-based constraining of the search space: *circle* would still have many meanings as well as other words, and building of this kind of system requires ontologies.

The ideal goal is to have a common-sense information retrieval and annotation system that all Web users can use efficiently, that can be applied to all domains, that accommodates relevance feedback and content-based analysis, and handles changes in time. While we are slowly moving towards this,

> we ontologize the world, and in return its sensible form is excavated, furrowed by categories that work at it and relate it from within, and spin forth the Web of significations that christen it, determine it, and fix it in the universe of discourse [50].

Acknowledgments

This thesis was instructed by professors Eero Hyvönen and Hannu Erkiö from University of Helsinki department of computer science. Appendix A was created in cooperation with graduate students Ilkka Autio and Jussi Lindgren from University of Helsinki department of computer science. Appendix B was inspected by professor Merja Salo from Helsinki University of Art and Design.

The development of the photograph exhibition system *Promotor* was initiated and led by Eero Hyvönen, and software was handed over to the UHe museum (http://www.halvi.helsinki.fi/museo/Homepage.htm) clients Kati Heinämies and Jaana Tegelberg. The software development team consisted of Samppa Saarela (architectural and visual design, implementation, Java), Kim Viljanen (scripts and semantic reasoning, Perl, Prolog), and Avril Styrman (ontologization, annotations). Promotion experts Tero Halonen and Tiina Metso made a great effort for the ontologization, as well as the UHe clients who also participated to the the ontologization and annotation processes.

The *Image Area Annotator Imarea* [77] was created in a Software Engineering Project (6 credit units) of UHe department of computer science. Marianne Korpela was the instructor of the software development team *Digimon* that consisted of Kai Hendry, Kimmo Könönen, Aleksi Mattila, Mirva Salminen, and Lasse Toimela. The clients of the project were Avril Styrman who specified the software, and Samppa Saarela.

I thank all who personally aided in different stages of creation of this thesis: Ilkka Autio, Hannu Erkiö, Aapo Halko, Tero Halonen, Heikki Helin, Tapani Hyttinen, Eero Hyvönen, Anne Isomursu, JP Kaljonen, Päivi Karimäki-Suvanto, Heikki Koivo, Matti Kääriäinen, Jussi Lindgren, Tiina Metso, Claus Montonen, Otto Nurmi, Kimmo Raatikainen, Vilho Raatikka, Pauliina Remes, Samppa Saarela, Merja Salo, and Kim Viljanen.

Due to the subject of the thesis many of the citations are on-line sources. Some citations are given both as literal sources and on-line sources. The functionality of the URI's has been verified in 11th of October 2004.

References

[1] AACR2, *Anglo-American Cataloging Rules*, 2nd Edition. http://www.nlc-bnc.ca/jsc/docs.html

[2] *Altavista search engine*: http://www.altavista.com

[3] ANSI, *American National Standards Institute*. http://www.ansi.org/

[4] AOL Search: http://search.aol.co.uk/

[5] Aristotle: *Categories*. Written circa 350 B.C.

[6] Ilkka Autio, Tapio Elomaa: *Flexible view recognition for indoor navigation based on Gabor filters and support vector machines.* Pattern Recognition vol. 36, nro. 12, p. 2769-2779, 2003.

[7] *ASCII, American Standard Code for Information Interchange*: http://www.asciitable.com/

[8] F. Baader, H-J. Bürckert, J. Heinsohn, J. Müller, B. Hollunder, B. Nebel, W. Nutt, H-J. Profitlich: *Terminological Knowledge Representation: A Proposal for a Terminological Logic*. DFKI Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, 1990. Updated version, taking into account the results of a discussion at the "International Workshop on Terminological Logics", Dagstuhl, May 1991.

[9] Taina Bagge: *Promootiot Helsingin Yliopistossa 1832-1967*. Publications of department of History of University of Helsinki nro 5. (Helsingin Yliopiston historian laitoksen julkaisuja N:o 5), Helsinki, 1974

[10] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David M. Blei, Michael I. Jordan: *Matching Words and Pictures*. In Journal of Machine Learning Research, vol. 3, nro. 2, p. 1107-1135, 2003.

[11] Ronald Barthes: *The Third Meaning*. Published in Image, Music, Text, p. 52-68. Translated by Stephen Heath. Hill and Wang, New York, 1995.

[12] Ronald Barthes: *La chambre claire*. Finnish edition by Kansankulttuuri Oy 1985, Gummerus Oy, Jyväskylä, 1985.

[13] Sean Bechhofer, Carole Goble, Ian Horrocks: *DAML+OIL is not Enough*, 2001. http://www.semanticweb.org/SWWS/program/full/paper40.pdf

[14] Walter Benjamin: *Taideteos teknisen uusinnettavuutensa aikakaudella* in "Messiaanisen sirpaleita. Kirjoituksia kielestä, historiasta ja pelastuksesta." Markku Koskinen, Keijo Rahkonen, Esa Sironen. Tutkijaliitto, Helsinki, p. 139-173, 1989. Originally published as "Das Kunstwerk im Zeitalterseiner technischen Reproduzierbarkeit" in 1936.

[15] Walter Benjamin: *Goethes Wahlverwandtschaften*. Edited by Hans-J. Weitz, Frankfurt a.M., Insel Verlag, p. 255-333, 1980. Originally published in 1922.

[16] J. van den Berg: *Subject retrieval in pictorial systems*. Proceedings of the 18th international congress of historical sciences, Montreal, Canada, p. 21-29, 1995.

[17] Tim Berners-Lee: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper, San Francisco, 1999.

[18] W. N. Borst, J. M. Akkermans: *Engineering Ontologies*. International Journal of Human-Computer Studies, vol. 46, nro. 2, p. 365-406, 1997.

[19] R. Boyer, S. Moore: *A fast string matching algorithm*. C.A.C.M, vol. 20, nro. 10, p. 767-772, 1977.

[20] R. Brachman, J. Schmolze: *An overview of the KL-ONE Knowledge representation System*. Cognitive Science, vol. 9, nro. 2, p. 171-216, 1985.

[21] D. Brickley, R. Guha: *RDF Schema specification 1.0, 2000*: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/

[22] C. Buckley, G. Salton: *Optimization of Relevance Feedback Weights*. In proceedings of SIGIR, 1995.

[23] *A list of Content-based Image Retrieval Systems*: http://www.sciences.univ-nantes.fr/info/perso/permanents/martinez/Researches/CBIRSs.html

[24] *CiteSeer search*: http://www.neci.nec.com/∼lawrence/citeseer.html
Citeseer browser: http://www.pmbrowser.info/citeseer.html

[25] *Princeton University Cognitive Science Laboratory*: http://www.cogsci.princeton.edu/

[26] Computer Vision Online Publications: http://www-2.cs.cmu.edu/∼cil/v-pubs.html

[27] *CYC, encyclopedia*. http://www.cyc.com/

[28] *Dublin Core Metadata Initiative*: http://dublincore.org/

[29] Michael Denny: *Ontology Building: A Survey of Editing Tools*, 2002. http://xml.com/pub/a/2002/11/06/ontologies.html

[30] DirectHit search: http://www.directhit.com/

[31] Description Logics. http://dl.kr.org/

[32] Johannes Clauberg: *Elementa philosophiae sive Ontosophiae*, 1647.

[33] DAML+OIL homepage: http://www.daml.org

[34] The Internet Encyclopedia of Philosophy: http://www.utm.edu/research/iep/

[35] J. English, M. Hearst, R. Sinha, K. Swearingen, K.-P.Lee: *Flexible search and navigation using faceted matadata*. Published in Communications of the ACM, volume 45, nro. 9, September 2002.

[36] H. Eriksson, R. W. Ferguson, Y. Shahar, M. A. Musen: *Automated Generation of Ontology Editors*. In Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW99), Banff, Alberta, Canada, October 16-21, 1999.

[37] *EuroWordNet* home page: http://www.illc.uva.nl/EuroWordNet/

[38] A. Farquhar, R. Fikes, J. Rice: *The Ontolingua Server*: A tool for Collaborative Ontology Construction, International Journal of Human-Computer Studies, vol. 46, nro. 6, p. 707-728, 1997.

[39] Christiane Fellbaum: *WordNet: An Electronic Lexical Database*, MIT Press, 1999.

[40] Dieter Fensel: *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer Verlag 2001.

[41] Dieter Fensel, R. Groenbroom: *Specifying Knowledge-based Systems with Reusable Components*. In Proceedings of 9th International Conference on Software Engineering and Knowledge Engineering (SEKE '97), Madrid, 1997.

[42] *FIPA, Foundation for Intelligent Physical Agents*. http://www.fipa.org/
List of agent platforms: http://www.fipa.org/resources/livesystems.html

[43] *Filosofisen tiedekunnan vuoden 2000 promootion muistokirja*. Helsingin yliopiston Filosofisen tiedekunnan promootiotoimikunta 2000. Edita, Helsinki, 2001.

[44] *Filosofisen tiedekunnan vuoden 1994 promootion muistokirja*. Helsingin yliopiston Filosofisen tiedekunnan promootiotoimikunta 1994. Helsinki-Hyvinkää, 1995.

[45] *Filosofisen tiedekunnan vuoden 1986 promootion muistokirja*. Helsingin yliopiston Filosofisen tiedekunnan promootiotoimikunta 1994. Yliopistopaino, Helsinki 1988.

[46] John Fiske: *Introduction to Communication Studies*. Second edition, Routledge, 1990. Finnish edition: *Merkkien kieli, johdatus viestinnän tutkimiseen*. Gummerus Oy, Jyväskylä, 1992.

[47] *Descriptive and Formal Ontology* homepage: http://www.formalontology.it/

[48] *FOTIOS*, a Dutch database system designed to manage and facilitate access to collections of photographic materials: http://www.knaw.nl/ecpa/sepia/workinggroups/wp5/fotios.html

[49] FOAF project homepage: http://www.foaf-project.org/

[50] Michel Foucault: *This is not a pipe*, translation of *Ceci n'est pas une pipe* that was originally published in 1973. Library of Congress Cataloging in Publication data, University of California Press, 1983.

[51] N. Fridman-Noy, C. D. Hafner: *The State of the Art in Ontology Design*. AI Magazine, vol. 18, nro. 3, p. 53-74, 1997.

[52] FRODO RDFSViz tool: http://www.dfki.uni-kl.de/frodo/RDFSViz/

[53] M. R. Genesereth, R. E. Fikes: *Knowledge Interchange Format*, Version 3.0, Reference Manual. Technical Report, Logic-92-1, Computer Science Dept., Stanford University, 1992: http://www.cs.umbc.edu/kse/kif/

[54] Sergei S. Goncharov: *Countable Boolean Algebras and Decidability*. Consultants Bureau, New York, 1997.

[55] *Google search*: http://www.google.com

[56] W. E. Grosso, H. Eriksson, R. W. Ferguson, J. H Gennari, S. W. Tu, M. A. Musen: *Knowledge Modeling at the Millenium*. (The Design and Evolution of Protégé-2000). In Proceedings of the twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW99), Banff, Alberta, Canada, October 16-21, 1999.

[57] Nicola Guarino: *Formal Ontology: Conceptual Analysis and Knowledge Representation*. International Journal of Human-Computer Studies, vol. 43, nro 2, p. 625-640, 1995.

[58] Nicola Guarino: *Formal Ontology in Information Systems*. IOS Press, Amsterdam, 1998.

[59] Nicola Guarino: *Review of Knowledge Representation: Logical, Philosophical, and Computational Foundations by John Sowa*. AI Magazine, vol. 22, nro. 3, p. 123-124, 2001. http://www.aaai.org/Library/Magazine/Vol22/22-03/Papers/AIMag22-03-019.pdf

[60] R. V. Guha: *Context Dependence of Representation in Cyc*. MCC Technical Report, p. 66-93, 1993.

[61] James R. Groff, Paul N. Weinberg: *SQL: The Complete Reference*. McGraw-Hill, Berkeley, California, 1999.

[62] T. R. Gruber: *Towards Principles for the design of Ontologies used for Knowledge Sharing*. International Journal of Human-Computer Studies, nro. 43, p. 907-928, 1995.

[63] Thomas R. Gruber: *A Translation Approach to Portable Ontology Specifications*. Knowledge Acquisition, nro. 5, p. 199-220, 1993.

[64] Rudolf Göckel: *Lexicon Philosophicum*, 1623.

[65] Ilkka Haikala, Jukka Märijärvi: *Ohjelmistotuotanto*, 6. painos. Suomen Atk-kustannus Oy, Gummerus Kirjapaino Oy, Jyväskylä, 1998.

[66] Tero Halonen: *Helsingin yliopiston matemaattis-luonnontieteellinen tiedekunta 150 vuotta*. Helsingin yliopisto, matemaattis-luonnontieteellinen tiedekunta, 2002.

[67] G. van Hejst, A. Schreiber, B. Wielinga: *Using Explicit Ontologies in KBS development*. International Journal of Human Computer Studies, 1997.

[68] On-line descriptions of promotions:
http://www.museo.helsinki.fi/Promotio.htm
http://www.halvi.helsinki.fi/museo/Historia.htm
http://www.helsinki.fi/promootiot/

[69] J. Hjelm: *Creating the Semantic Web with RDF*. John Wiley and Sons, New York, 2001.

[70] T. Hoffmann. *Learning and representing topic. A hierarchical mixture model for word occurrence document in document databases.* In workshop on learning from text and the web, CMU, 1998.

[71] HotBot search: http://www.hotbot.com/

[72] Eero Hyvönen, Avril Styrman, Samppa Saarela: *Ontology-Based Image Retrieval*. In Towards the Semantic Web and Web Services, Proceedings of XML Finland 2002 Conference. pp. 15-27. HIIT Publications, 2002-03. http://www.cs.helsinki.fi/u/eahyvone/publications/yomuseum.pdf

[73] Eero Hyvönen, Samppa Saarela, Kim Viljanen: *Ontogator: Combining View- and Ontology Based Search with Semantic Browsing*. In proceedings of XML Finland 2003, Kuopio, Finland, 2003. http://www.cs.helsinki.fi/u/eahyvone/publications/xmlfinland2003/yomXMLFinland2003.pdf

[74] E. Hyvönen, S. Saarela, K. Viljanen, E. Mäkelä, A. Valo, M. Salminen, S. Kettula, M. Junnila: A Cultural Community Portal for Publishing Museum Collections on the Semantic Web. Proceedings of 16th European Conference on Artificial Intelligence (ECAI2004), Workshop on Application of Semantic Web Technologies to Web Communities, Valencia, Spain, 2004. CEUR Workshop Proceedings, Vol 107, 2004

[75] *ICOM, International Council of Museums*: http://www.cidoc.icom.org/

[76] *ICONographic CLASsification System*: http://www.iconclass.nl/

[77] Image Area Annotator. Software development group homepage: http://www.cs.helsinki.fi/group/digimon/

[78] INOA business search: http://www.inoa.fi/

[79] *ISAD(G), General International Archival Standard*: http://www.mclink.it/personal/MD1431/sito/isaargrp/isad(g)e.html http://www.ica.org/biblio/cds/isad_g_2e.pdf

[80] *ISBN, International Standard Book Number*: http://www.isbn.org/

[81] *ISO*, International Organization for Standardization: http://www.iso.ch/iso/en/ISOOnline.frontpage

[82] J. Jannik, S. Pichai, D. Verheijen, G. Wiederhold: *Encapsulation and composition of Ontologies*. In Proceedings of the AAAI '98 Workshop on AI and Information Integration, Madison, WI, July 26-27, 1998.

[83] *JAVA programming language*: http://java.sun.com

[84] *JENA*: http://www.hpl.hp.com/semweb/doc/tutorial/index.html

[85] Nicole Joliceur: *Charcot : Deux concepts de nature.* Montréal, Qc, Artexte, 1988.

[86] Kalervo Järvelin: *Tekstitiedonhaku tietokannoista (Text retrieval in databases).* Asiantuntija-sarja: Tiedonhaku, Suomen ATK-kustannus Espoo, Finland, 1995.

[87] *KIF, Knowledge Interchange Format*: http://logic.stanford.edu/kif/kif.html

[88] M. Kifer, G. Lausen, J. Wu: *Logical Foundations of Object-Oriented and Frame-Based Languages.* Journal of the ACM, nro. 42, 1995.

[89] Edwin Klijn, Yola de Lusenet: *In the Picture.* Preservation and digitalization of European photograohic collections. European Commission on Preservation and Access, Amsterdam, 2000.

[90] Matti Klinge, Rainer Knapas, Anto Leikola, John Strömberg: *Helsingin Yliopisto 1640-1990*, kolmas osa 1917-1990, Otava, Keuruu, 1917-1990.

[91] S. Klink: *Query reformulation with collaborative concept-based expansion.* Proceedings of the First International Workshop on Web Document Analysis, Seattle, WA, 2001.

[92] Matti Klinge: *Kuninkaallinen Turun akatemia 1640-1808. Helsingin yliopisto 1640-1990*, ensimmäinen osa. Keuruu, 1987.

[93] D. E. Knuth, J. H. Morris, V. R. Pratt: *Fast pattern matching in strings.* SIAM Journal on Computing, p. 323-350, June 1977.

[94] Simo Heininen, Erkki Kansanaho, Kauko Pirinen, Martti Parvio, Markku Heikkilä: *Kutsu Helsingin Yliopiston Teologisen Tiedekunnan Tohtoripromootioon toukokuun 23. päivänä 1997*, Oy Edita Ab, Helsinki, 1997.

[95] Ora Lassila, R. Swick (editors): *Resource Description Framework (RDF)*: Model and syntax specification. Technical report, 1999. W3C Recommendation. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[96] Ora Lassila and Deborah McGuinness: *The Role of Frame-Based Representation on the Semantic Web.* Knowledge Systems Laboratory Report. KSL-01-02, Stanford University, 2001.

[97] V. Lavrenko, R. Manmatha, J. Leon: *A Model for Learning the Semantics of Pictures.* Preproceedings of NIPS 2003.
http://books.nips.cc/papers/files/nips16/NIPS2003_AA70.pdf

[98] LCSH, Library of Congress Subject Headings: http://www.carl.org/tlc/crs/shed0014.htm

[99] K.-P. Lee, K. Swearingen, K. Li, M. Hearst. Faceted metadata for image search and browsing. Proceedings of CHI 2003, April 5-10, Fort Lauderdale, USA. Association for Computing Machinery (ACM), USA, 2003.

[100] D. B. Lenat, R. V. Guha: *Building large knowledge-based systems.* Representation and inference in the Cyc project, Addison-Wesley, Reading, Massachusetts, 1990.

[101] Gunther Lenz: *.NET - A Complete Development Cycle*, Addison-Wesley, 2003.

[102] D. Lin: *Using collocation statistics in information extraction.* Proceedings of the Seventh Message Understanding Conference (MUC-7), San Francisco, CA, 1998.

[103] Hugo Liu, Henry Lieberman. *Robust photo retrieval using world semantics*. Proceedings of the 3rd International Conference on Language Resources And Evaluation Workshop: Creating and Using Semantics for Information Retrieval and Filtering: State-of-the-art and Future Research (LREC2002), pp. 15-20, Las Palmas, Canary Islands, 2002.

[104] Jacob Lorhard: *Theatrum Philosophicum*, 1613.

[105] Ye Lu, Chunhui Hu, Xingquan Zhu, HongJiang Zhang, Qiang Yang: *A Unified Framework for Semantics and Feature Based Relevance Feedback in Image Retrieval Systems*. Proceedings of ACM MM 2000, p. 31- 38, 2000.

[106] Lycos search: http://sjc-search.sjc.lycos.com/

[107] S. McIlraith, T. Son, H. Zeng: *Semantic web services*. IEEE Intelligent Systems, p. 46-53, March/April 2001.

[108] Deborah L. McGuinness: *Description Logics Emerge from Ivory Towers*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-08 2001. In the Proceedings of the International Workshop on Description Logics. Stanford, CA, August 2001.

[109] Tiina Metso: *Tapaillaan. Akateemisia juhlia ja tapoja.* Yliopistopaino, 1998.

[110] Ray Monk: *Russell; Mathematics: Dreams and Nightmares*. Lennart Sane Agency AB, 1997.

[111] Netscape search: http://search.netscape.com

[112] Ilkka Niiniluoto: *Johdatus Tieteenfilosofiaan*. Kustannusosakeyhtiö Otavan painolaitokset, Keuruu, 1997.

[113] Ian Niles, Adam Pease: *Origins of the IEEE standard upper ontology*. Technical report, Teknowledge, Palo Alto, 2001.

[114] *OIL, Ontology Interchange Language*: http://www.ontoknowledge.org/oil/

[115] *ONTOLINGUA*: http://www.ksl.stanford.edu/software/ontolingua/

[116] *Open Directory Project*: http://dmoz.org/about.html

[117] *OWL* Web Ontology Language 1.0 specification: http://www.w3.org/TR/2002/WD-owl-ref-20020729/

[118] T.Petersen: *Introduction to the Art and Architecture Thesaurus*. Oxford University Press, 1994: http://www.getty.edu/research/tools/vocabulary/aat/

[119] H.J. Peat, P. Willett: *The limitations of term co-occurrence data for query expansion in document retrieval systems*. Journal of the ASIS, Vol. 42, nro. 5, p. 378-383, 1991.

[120] The source for Perl, development and conferences: http://www.perl.com/

[121] Th. Pirlein, R. Studer: *Integrating the Reuse of Commonsense Ontologies and Problem-Solving Methods*. International Journal of Expert Systems: Research and Applications, 1999.

[122] Plato: *Republic* 509d ff.

[123] A. S. Pollit: *The key role of classification and indexing in view-based searching*. Technical report, University of Huddersfield, UK, 1999. http://www.ifla.org/IV/ifla63/63polst.pdf

[124] *Protégé-2000*: http://protege.stanford.edu/index.html

[125] Prolog: http://www.xml.com/pub/a/2001/04/25/prologrdf/

[126] A. R. Puerta, J. W. Egar, S. W. Tu, M. A. Musen: *A Multiple-method Knowledge-Acquisition Shell for the Automatic Generation of Knowledge-acquisition Tools*. Knowledge Acquisition, vol. 4, nro. 2, p. 171-196, 1992.

[127] Dave Beckett, editor: *RDF/XML Syntax Specification (Revised)*. W3C Recommendation, 10th of February, 2004.
http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[128] *RDF Semantics*, W3C Recommendation, 10 February 2004.
http://www.w3.org/TR/rdf-mt/

[129] RDF Validation service by W3C: http://www.w3.org/RDF/Validator/

[130] *RDQL Tutorial*: http://www.hpl.hp.com/semweb/doc/tutorial/RDQL/

[131] Rudolf RDFViz tool: http://www.ilrt.bris.ac.uk/discovery/rdf-dev/rudolf/rdfviz/

[132] Stuart Russell, Peter Norveig: *Artificial Intelligence: A Modern Approach*. Addison-Wesley, 1994.

[133] G. Salton, M. J. McGill: *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

[134] A.Th. Schreiber, Barbara Dubbeldam, Jan Wielemaker, Bob Wielinga: *Ontology-Based Photo Annotation*. IEEE Intelligent Systems, 16:66-74, 2001.

[135] *Semantic Web Community Portal*: http://www.semanticweb.org/

[136] Pia Sivenius: *Mestari ja Marguerite*. Published in magazine Nuori Voima, nro. 6, 1995.

[137] Barry Smith: *Ontology and information systems*, forthcoming in Stanford Encyclopedia of Philosophy.

[138] Barry Smith, Christopher Welty: *Ontology: Towards a New Synthesis*, introduction to proceedings of 2nd international conference on Formal Ontology and Information Systems, New York, ACM Press, 2001. http://www.cs.vassar.edu/faculty/welty/papers/fois-intro.pdf

[139] SONERA's topic search: http://www.fi/CDA.CompanySearch.Companies/

[140] R. Studer, H. Eriksson, J. H. Gennari, S. W. Tu, D. Fensel, M. Musen: *Ontologies and the Configuration of Problem-Solving Methods*. In B. R Gaines and M. A. Musen (eds.), Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, 1996.

[141] John F. Sowa: *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, 1984.

[142] John F. Sowa: *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

[143] John F. Sowa's Web pages: *http://www.jfsowa.com/*

[144] Peter Suber: *Glossary of First-Order Logic Peter Suber, Philosophy Department, Earlham College*: http://www.earlham.edu/∼peters/courses/logsys/glossary.htm

[145] *SUO, Standard Upper Ontology*: *http://suo.ieee.org*

[146] *Scalable Vector Graphics SVG*: http://www.w3.org/Graphics/SVG/

[147] B. Swartout, R. Patil, K. Knight, T. Russ: *Toward Distributed use of Large-scale Ontologies*. In B. R. Gaines and M. A. Musen, (eds.), Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, 1996.

[148] *Semantic Web Web Rule Language*: http://www.daml.org/2003/11/swrl/

[149] *URI Uniform Resource Identifier* specifications: http://www.w3.org/Addressing/

[150] M. Uschold, M. Grüninger: *Ontologies, Principles, Methods and Applications*. Knowledge Engineerin Review, vol. 11, nro. 2, 1996.

[151] Historical review of View-Based Systems: http://www.view-based-systems.com

[152] John Wilkins: *An Essay Towards a Real Character and A Philosophical Language*. Originally published in 1668. http://reliant.teknowledge.com/Wilkins/

[153] E. Voorhees: *Query expansion using lexical-semantic relations*. Proceedings of ACM SIGIR Intl. Conf. on Research and Development in Information Retrieval, p. 61-69, 1994.

[154] S. Weibel, J. Gridby, E. Miller: *OCLC/NCSA Metadata Worksop Report*. Dublin, EUA, 1995

[155] B.J. Wielinga, A.Th. Schreiber, J. Wielemaker, J.A.C. Sandberg: *From Thesaurus to Ontology*. University of Amsterdam, Social Science Informatics.

[156] *Visual Resources Association*: http://www.vraweb.org/
VRA Core Categories, version 3.0: http://www.vraweb.org/vracore3.htm

[157] *WordNet*: http://www.cogsci.princeton.edu/∼wn/

[158] *W3C, World Wide Web Consortium*: http://www.w3.org/

[159] *XML*: http://www.w3.org/XML/

[160] J. Xu, W.B. Croft: *Query Expansion Using Local and Global Document Analysis*. Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 4-11, 1996.

[161] *YSA, Yleinen suomalainen asiasanasto*, verkkopalvelu: http://vesa.lib.helsinki.fi/ysa/index.html

[162] W. Zhao, R. Chellappa: *Image Based Face Recognition, Issues and Methods*. Image Recognition and Classification, Edited by B. Javidi, Mercel Dekker, p. 375-402, 2002: http://www.cfar.umd.edu/∼wyzhao/publication.html

Appendix A: Content-Based Image Retrieval CBIR

While the text-based, field-based, and structure-based image retrieval paradigms (sect. 2) analyze the metadata that is created by human annotators, the content-based image retrieval (CBIR) paradigm analyzes the image data itself, i.e., the physical elements of images that were defined in section 1.2. Figure 39 depicts a hierarchy of **Image understanding and recognition**. The **Computerized** part is explained in this appendix, and the **Not computerized** part is explained in appendix B.

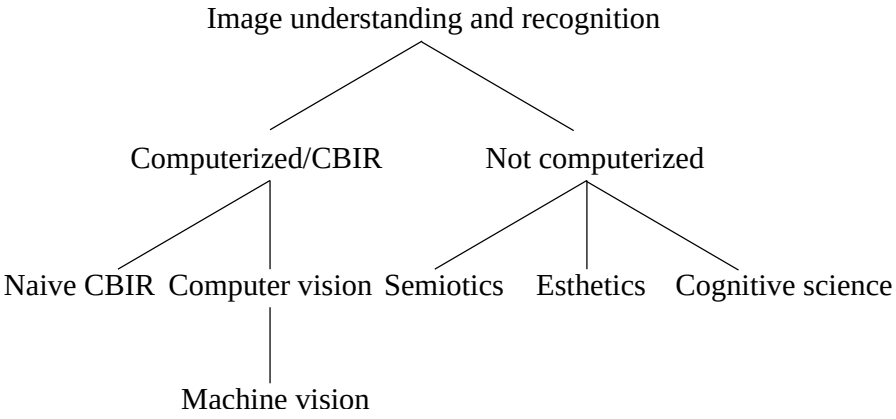


Figure 39: Some categories that describe fields of image understanding and recognition.

Naive CBIR

Naive CBIR techniques collect different kinds of statistics from digital images such as color and edge distribution. In other words, different filters are applied to images and the filters' output is analyzed to find the kinds of regularities that can be used in categorizing images. Nowadays for example search engines AltaVista, Yahoo and several others [23] accommodate naive CBIR-techniques in image retrieval.

This branch of image understanding and recognition is called here naive because the 'understanding' of the approach is very raw: two images are similar if they have the same relative brightness or amount of the same color. However, these techniques can be used as components with other more sophisticated techniques which are discussed in the following.

Computer vision

Computer vision includes all the computerized methods and theories that aim to create applications that can automatically distinguish a somewhat deeper meaning of images' physical elements than naive CBIR. There can be seen a similar kind of relation between standardized machine vision techniques and yet unstandardized computer vision techniques than between 'normal' applications and Artificial Intelligence applications: when a new computer vision technique becomes standardized, i.e., it is sure that it works well, then it can be called a machine vision technique.

Computer vision aims to do automatically what can be done manually with tools like *Imarea* [77], a prototypical editor that can be used to create metadata that connects images' physical elements to abstract elements. The user can specify any coordinates of 2D images' with the basic image processing tools: quadrangle, polygon, and freehand-tool. The specified coordinates can then be described with the aid of different metadata schemas: Dublin core [28], FOAF [49], and WordNet [157]. The created metadata can be saved either as a separate file or into the metadata field of JPEG and PNG image formats in a reusable XML-based format (sect. 3.2.2) that follows the SVG-standard [146].

Figure 40 depicts a view of Imarea's user interface. As can be seen in the category on

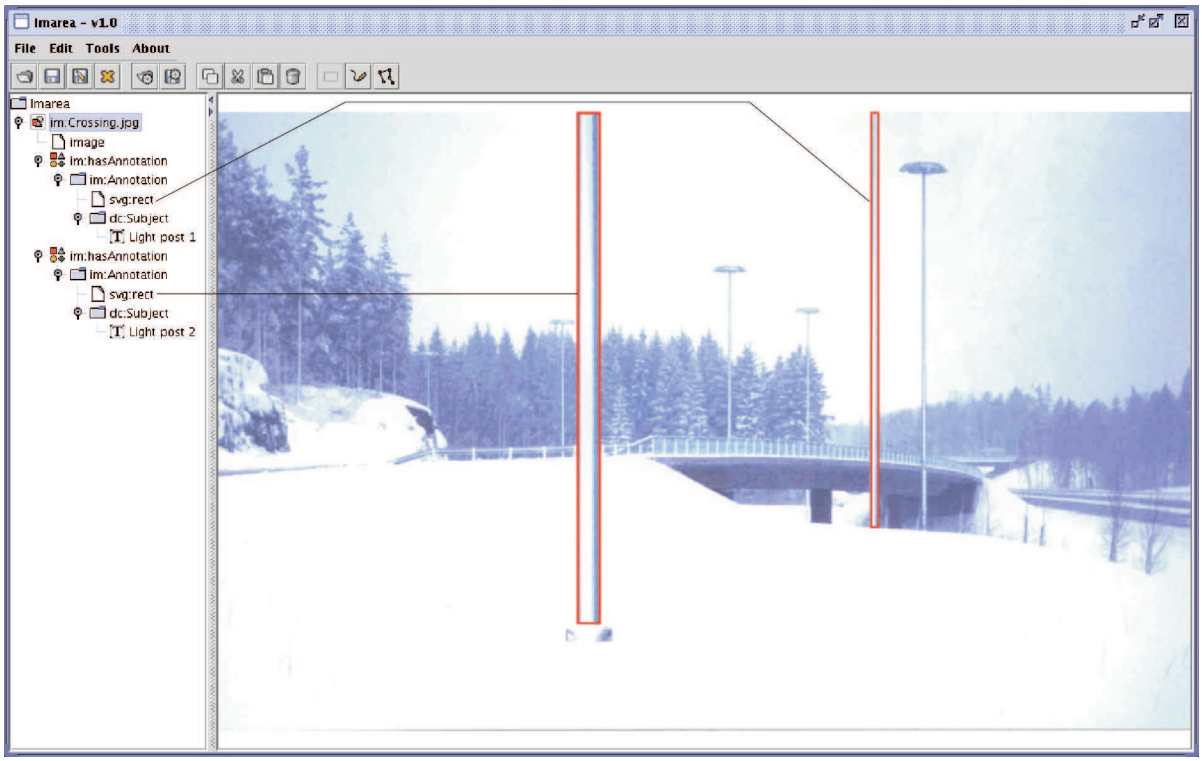


Figure 40: A view of *Image Area Annotator*'s user interface.

the left side of the figure, an image with file name 'Crossing.jpg' has two annotations: coordinates of the two light posts have been specified with rectangles (svg:rect), and the rectangles have been linked to Dublin Core's attribute *Subject* with textual values 'Light post 1' and 'Light post 2'.

Machine vision

Machine vision is widely used in the industry. A simple example of machine vision is a pipeline-application where a robot identifies 'bad' oranges from within the ones that are good enough for selling. The robot is equipped with a *camera*, a *model* of an average 'good' orange, a model of an average 'bad' orange, and a *comparison algorithm*. The models[42] are taught to the robot with a *machine learning application* by providing a

[42]Several techniques can be used in the model construction such as Bayes and neural networks, fuzzy logic, self-organizing maps, and combinations of these. However, hard-coded sensors can be applied successfully in recognizing certain types of objects in known conditions without using modern machine learning techniques. In these cases the 'learning' is done by adjusting the sensors.

sufficient *training set*; a set of photos about good and bad oranges. When the oranges are moving on the pipeline the robot takes a photo of every single orange. The photo is filtered into a form that can be used by the comparison algorithm to decide whether to accept an orange or not. The known and unchanging conditions in the pipeline environment enable a successful filtering and comparison in real-time. There is always a fixed *amount of light*, a fixed *posing angle*, a fixed *size* of an orange relative to the size of the whole photograph, and it is naturally know what is being recognized.

Towards integrating formal ontologies with CBIR

The problems that are solved by the stable conditions in machine vision are the fundamental problems in other fields of computer vision [6, 162, 26]. Take an arbitrary photo about a human as an example. There can be different lighting conditions, different posing angles, and different sizes of the subject relative to the size of the whole photo. The filtering should dispose irrelevant information from the photos and it is difficult to decide which part of the information is relevant or irrelevant because of the heterogeneous conditions. In the general case the problems of lighting, size, and pose cannot be cleanly separated. Object o_1 under lighting conditions l_1 may look exactly like object o_2 under lighting conditions l_2. With an arbitrary image there is no way to decide even in principle if the object that is the subject of recognition is o_1 or o_2. As an example of complexity, an object can have n possible posing angles, m possible sizes, and p possible lighting conditions. The size of a model containing all this information would be $n \cdot m \cdot p$, when the size of a model that could recognize the same object in a pipeline environment would be 1.

Computer vision articles give often a very optimistic idea about some methods used in practice. Applications succeed to work sufficiently and even well in constrained environments or domains such as in corridors of a house and in face recognition, but once they step out of the familiar environment the recognition abilities face an insuperable wall. Applications can recognize only what is included in their model that is learned by analyzing the training set photos. If an application can recognize a cow, it does not help in recognizing a human. If both cows and humans were to be recognized there should be two models, one for cows and one for humans. This indicates that the size of a model that could recognize more than one type of objects would grow linearly relative to the amount of objects that can be successfully recognized.

The fundamental goal of computer vision is to create a system that can use something that it has already learned to learn new things more easily. The eventual solution to the learning problem lies in combining CBIR techniques with well-formed information structures, formal ontologies. Interpretation of any image is context dependent and it is clear that understanding the relations of objects facilitates recognition. Recently, semantically oriented techniques have been used successfully with CBIR techniques [10, 97, 103]. For example in [10] the metadata that surround an image, such as text on a Web page where an image situated, is used as a starting point to recognize the image's physical elements. A *cluster* model similar to the category tree is used to reason that a wave belongs to the sea and the sea belongs to the landscape for example. Semantic methods [70] were used in constructing the cluster model.

The benefits of the ontology-based approach are explained with a greatly simplified example. Let there be n objects: $o_1, o_2, \ldots, o_n$. Let $\mathcal{O}$ denote the set of all these objects: $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$. The model that recognizes o_1 is denoted as m_1, the model that

recognizes o_2 as m_2, and so forth, the model that recognizes all objects in $\mathcal{O}$ is denoted as $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$. Let the size of each individual model $m_k (k \in 1, \ldots, n)$ be 1. Therefore the size of $\mathcal{M}$ is n, and the size of $\mathcal{M}$ grows linearly relative to the amount of objects that can be recognized.

The goal is to identify and determine what is common to all members of $\mathcal{M}$ and $\mathcal{O}$ so that the things that have already been learned would facilitate recognizing new things of the same type. An intersection is performed between all members of $\mathcal{M}$, which is denoted by $\mathcal{M}_\cap = \cap_{k=1}^{n} m_k$. $\mathcal{M}_\cap$ contains all that is common to all members of $\mathcal{M}$. When $\mathcal{M}_\cap$ has been calculated, $\mathcal{M}_\cap$ can be dispatched from every member of $\mathcal{M}$ because there is no need to multiply data. This is done by calculating the difference between all members of $\mathcal{M}$ and $\mathcal{M}_\cap$: $\mathcal{M} \backslash \mathcal{M}_\cap = \{m_1, m_2, \ldots, m_n\} \backslash \mathcal{M}_\cap = \{m_1 \backslash \mathcal{M}_\cap, m_2 \backslash \mathcal{M}_\cap, \ldots, m_n \backslash \mathcal{M}_\cap\}$. The size of $\mathcal{M} \backslash \mathcal{M}_\cap$ together with $\mathcal{M}_\cap$ is smaller than $\mathcal{M}$ if two or more members of $\mathcal{M} have anything in common$: $size(\mathcal{M} \backslash \mathcal{M}_\cap) + size(\mathcal{M}_\cap) < size(\mathcal{M})$.

As a concrete example, let $\mathcal{M}$ consist of $horse$, $zebra$, and $mule$, i.e., model $\mathcal{M}$ is capable of recognizing these three animals: $\mathcal{M} = \{horse, zebra, mule\}$, and $size(\mathcal{M}) = 3$. The intersection $\mathcal{M}_\cap = horse \cap zebra \cap mule$ contains all that is common to all members of $\mathcal{M}$: they all walk on four legs, all have approximately the same kind of relative proportions of body parts, and all have a mane, ears, eyes, and tail amongst other features. Members of the difference $\{horse, zebra, mule\} \backslash \mathcal{M}_\cap$ then again contain only the special features of horses, zebras, and mules such as the size factor, special form of the mane, ears, and tail, and possible colors of the coat. If the size of the original model M was 3, and the size of $\mathcal{M}_\cap$ is 0.5, then the size of the new model $\mathcal{M} \backslash \mathcal{M}_\cap$ together with $\mathcal{M}_\cap$ is $size(\mathcal{M}) - n \cdot size(\mathcal{M}_\cap) + size(\mathcal{M}_\cap) = 3 - 3 \cdot 0.5 + 0.5 = 2$. The greater is the amount of items that have similarities, the greater is the relative benefit because all things are related up some degree.

Automatic clustering takes place at this point. The actual clustering of the models that have been created is a great problem because the model frameworks that are used today are not efficiently structured, and so it cannot be said just where in a model is the part that recognizes a certain thing. If the models were well-structured, the clustering problem could be eventually reduced into matching sequences or sets of integers, in order to form a hierarchy of these sets. If the models are not semantically oriented in any way, they can be considered as sets of large integers without no indication of the reason why the integers are there as they are, and a successful comparison of two sets would seem to be quite impossible.

The recognition of the members of $\mathcal{O}$ walks hand in hand with the clustering. Formal ontologies are inevitably a promising aid to this problem: understanding the relations of members of $\mathcal{O}$ helps understanding the relations of members of $\mathcal{M}$ and vice versa.

Appendix B: General Concepts of Image Analysis

This appendix briefly discusses the **Not computerized** (fig. 39 on p. 78) part of image understanding and recognition, while appendix A explained the **Computerized** part (fig. 39 on p. 78). The source of this appendix is [46] if other sources are not explicitly mentioned.

A set of terms is used in analyzing images and art in general. The specification of these terms is vague and they describe more or less the things. Some of these are explained with *physical* and *abstract elements*, and their *resemblances* that were defined in section 1.2. The things that are not a part of an image but have affect on the interpretation, such as artists' intentions and history, can be taken also as abstract or physical elements. The table below reveals the terms that are explained in the following. The abstract elements are divided into direct and indirect resemblances of physical elements in the table.

Physical Elements	**Direct Resemblances**	**Indirect Resemblances**
Icon & Index	Icon	Index
Factual Content	Truth content	Truth content
Sign	Signification	Signification
Denotation	Denotation	Connotation
Punctum & Studium	Punctum & Studium	Punctum & Studium

Icon – Index

The clearest examples of icons are abstract universal forms such as 'square' or 'triangle' that are in *iconic resemblance* (IR). IR can be founded on Aristotle's *correspondence theory of truth* [112] which states that truth is a kind of correlation between belief and reality; a sentence is true if it corresponds the reality. IR is not crisp as in mathematical logic, but it is always fuzzy and subjective. This is realized with the following test that measures the degree of IR.

An arbitrary image is denoted by i. A million people[43] are asked "what is the single strongest form or idea in i?" The average answer is a. Then another million people are given a task to draw a. The closer i is to an average drawing d, the stronger is the IR of i. This way the test of IR can be seen as a triple (i, a, d), where the similarity of i and d is being measured. The IR would probably be quite strong if i would be either the left or the right side of figure 41, but if i would be the whole figure 41, the IR would not be as strong. The direct resemblances 'circle' and 'square' would mediate together and resemble something else, whereby i would probably not be very close to d

Figure 41: ?

[43] Or an adequate amount of people in order to make the results of the test objective.

because there is not a single right word, phrase or concept that describes the indirect resemblances of the whole figure 41. In this case i could resemble 'circle and square', 'geometrical objects', or 'two forms' for example. The same pattern can be distinguished with myriads of things like alphabetic letters: one letter alone can resemble itself but a collection of letters forms other concepts, words.

In the above test the average answer a was in form of text, but the answer could also be in form of sound or anything else that can be percepted. Also i and d could be of any perceptual type, other than a, because if a is of the same type than i and d, the test of IR can get blurred. The test can be applied also in measuring the similarity of two or more things of different perceptual types. As an example, word 'pipe' is the average answer a when i is an image about a pipe. Image of pipe and a real physical smoking pipe are in iconic resemblance because the average answer a is the same for both of them.

Unlike icon, *index* does not have a strong iconic resemblance. Index resembles primarily something else than its own visual form. Icon-index relationships of the images on figure 42 and are analyzed in the following. The physical elements resembling smoke on the

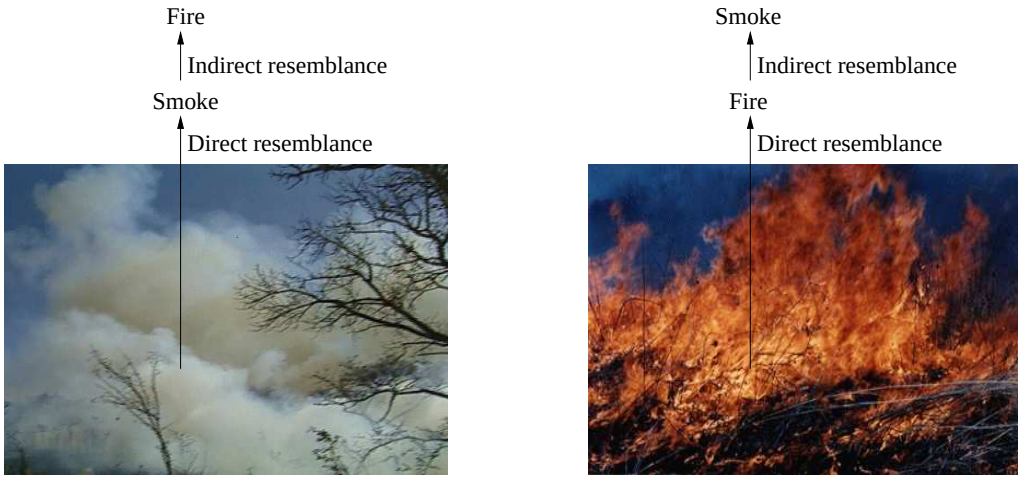


Figure 42: Two separate photographs.

left side of figure 42 can be interpreted as index of fire because smoke resembles fire indirectly. Fire can be considered as index of smoke on the right side of figure 42 but not as strongly as smoke is an index of fire: smoke could be easily interpreted also as fog or cloud for example. When fire and smoke clearly appear in the same photo, a mapping can be made between these two direct resemblances.

The general usage of icon and index is tied to physical elements. If abstract relations of things are described with icon and index outside of any physical elements, two more terms have to be defined: *abstract icon* and *abstract index*. In the left of figure 42 smoke is an index of an abstract icon, fire, and in the right side fire would be an index of an abstract icon, smoke. To avoid these complex terms we can as well use *abstract elements* that are derived from direct and indirect *resemblances* of *physical elements*.

The general usage of icon and index can also be specified with a simple logical predicate $resembles(b, c)$, meaning 'physical element b resembles abstract element c'. If the predicate is extended so that b and c could be both physical and abstract elements, it could be used also to describe relations of pure abstract ontological concepts instead of just linking images' physical elements to abstract concepts.

Factual Content – Truth Content

Walter Benjamin (1892-1940) used term *factual content* (translation from German) [15] for the physical elements of a work of art and *truth content* to describe the abstract resemblances of the factual content. When truth content is a direct resemblance, then the factual content is in iconic resemblance. When truth content is an indirect resemblance, it can be derived from the factual content with a chain or network of two or more resemblances. Benjamin also discussed the effect of the environment to the interpretation of a work of art along the time [14]: the truth content can change along the change of time and place while the factual content remains relatively the same.

Sign – Signification

Signs can be percepted in audial, visual, or in any other sensible form. These signs have meanings, *significations*. In images signs are physical elements and significations are abstract elements that are derived from the direct and indirect resemblances of the physical elements.

Denotation – Connotation

Denotation and connotation are not constrained only to the analysis but fit describing also the photographing process (creation of art) where denotation stands for 'what is photographed' and connotation for 'how was it photographed' [46, 11]. In the analysis denotation stands for the physical elements that a piece of art consists of, and their direct resemblances. When physical element p is denotative, p is in iconic resemblance. Connotative physical elements then again resemble something else than their own visual form, like smoke resembles fire. The meaning of the connotative physical elements can be eventually derived from their resemblances and how these act with the resemblances of other physical elements.

Punctum – Studium

Roland Barthes (1915-1980) defined punctum [12] as something that clearly comes out of the image, the strongest emotion, Firstness (sect. 3.1.4). Studium is the objective study that explains how and where the punctum is derived from. Barthes maintained that "It is impossible to set a deterministic rule to specify the relationships that connect punctum to studium in general," which is true because subjective opinions of individual humans cannot be deterministically known or anticipated, at least up to a degree of 100%. The average punctums and studiums can however be measured by asking these from many people, as was done with the test of iconic resemblance. As a physical element punctum denotes some area of an image, i.e., the whole image or a proper subset of the image, when studium would explain its direct and indirect resemblances. As an abstract element punctum is derived from the physical elements directly or via a path of resemblances. If punctum is a direct resemblance, then studium would explain that the punctum is in iconic resemblance. If punctum is an indirect resemblance, then studium would explain the paths of resemblances between the physical elements and the punctum.

Appendix C: Examples of Ontologies

This appendix gives examples of well known philosophical and formal ontologies. The examples are given in a timely order based on the appearance of the original sources. The category tree examples are in accordance with the definition of section 3.1 with one exception that is examined and then represented as a plain category tree structure. The absurd type $\bot$ is left undrawn in most of the cases because of representational reasons. Three of today's largest existing formal ontologies, the Open Directory Project, WordNet, and CYC are also briefly reviewed.

The table below connects some of the top level categories of the category tree examples, i.e., tells which categories in different trees have approximately the same meaning.

Plato	**Porphyry**	**Brentano**	**CYC**	**Sowa**
$\top$	**Substance**	**Being**	**Thing**	$\top$
Physical	**Material**	**Substance**	**?**	**Physical**
Abstract	**Immaterial**	**Accident**	**?**	**Abstract**

Plato's Categories

The first example reveals Plato's (427-347 BC) categories according to [122]. Plato divided things generally to two categories, to the **Abstract** world as the ultimate truth, and to the **Physical** world as merely an imperfect implementation of the abstract world. In fact, Plato gave the lowest value to **Image**s, because images (such as statues and paintings) are only reflections of the **Physical** world. Therefore, **Image** and **Physical objects** could have been set directly below $\top$ as well. Plato named the relation between **Image**s and the **Physical** world as *mimesis*, and the relation between the **Abstract** and the **Physical** world as *mathesis*.

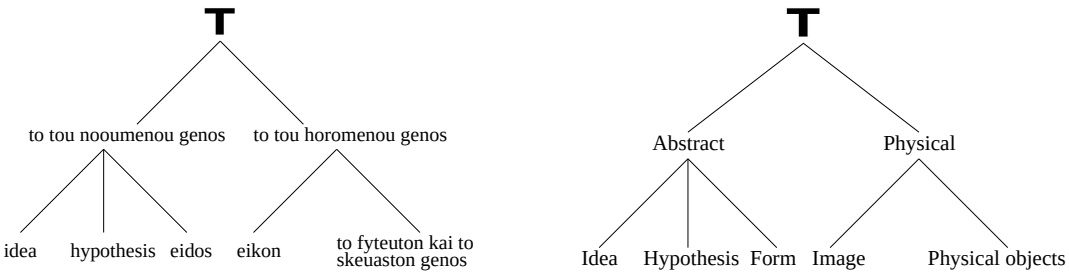


Figure 43: Plato's categories in Greek and in English.

Even though Plato preceded Aristotle (384-322 BC) also in metaphysics, Plato's works were widely studied in the Latin speaking Europe very much later than Aristotle's works on logic [5], which were translated by the Neoplatonist Boethius (c. 480-524AD) and were studied already during the Roman Empire. Plato's 'ultimate truth' corresponds to category **Abstract**, because he did not explicitly specify the universal type $\top$ which is drawn on figure 43. Therefore, Plato's categories could be represented also simply by **Abstract** and its three subcategories, and by setting the physical objects as instances of one or more successor of **Abstract**. **Image**s could then be set as (sub)instances of the physical objects.

Tree of Porphyry

The oldest known tree diagram that depicts a philosophical ontology was drawn by the Greek philosopher Porphyry (circa 233-305 AD) in his *Introduction to Categories*, which is a commentary on Aristotle's *Categories*, that is again the first part of Aristotle's *Logic* [5]. It incorporated Aristotle's logical foundations into Neoplatonism, and especially the problem of universals and particulars[44].

Figure 44 depicts John Sowa's clarification [142, 143] of the Tree of Porphyry as it was drawn by the logician Peter of Spain in 1329. It illustrates the categories under **Substance**, which is called the supreme genus or the most general category. Naturally, Aristotle's **Substance** corresponds to the universal type $\top$. The tree also visually introduced, or made a separation between the categories and the *individuals* (*instances*, *particulars*).

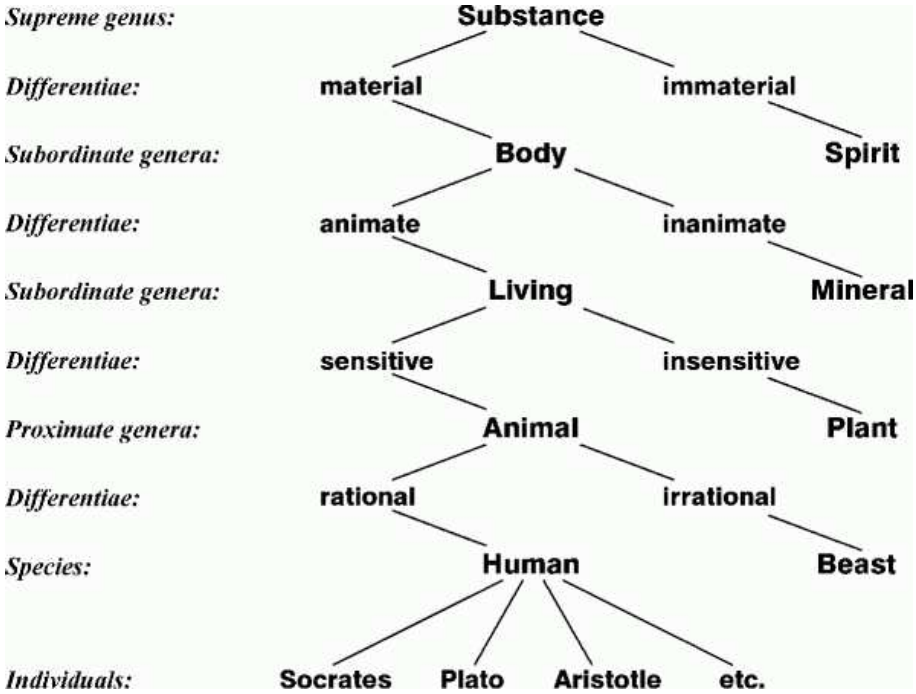


Figure 44: The tree of Pophyry.

This visualization of Aristotle's categories does not follow exactly the definition of the category tree, but the same information can be represented with plain category tree formalism as can be seen in figure 45 in p. 87, that is even simpler than the Tree of Porphyry. And after all, the Tree of Porphyry is only a visualization of Aristotle's categories.

[44] *Boethius' Isagoge* is a Latin translation of Porphyry's *Introduction to Categories*. *Boethius' Isagoge* became a standard medieval textbook that set the stage for medieval philosophical-theological development of logic and the problem of universals.

The category trees in figure 45 are all in correspondence with the Tree of Porphyry. The viewer can reason the meaning of the categories even if they are not visually classified with explicit relation-tags as in the Tree of Porphyry. Also the division to categories and individuals is clear, even if there are no explicit tags that tell this.

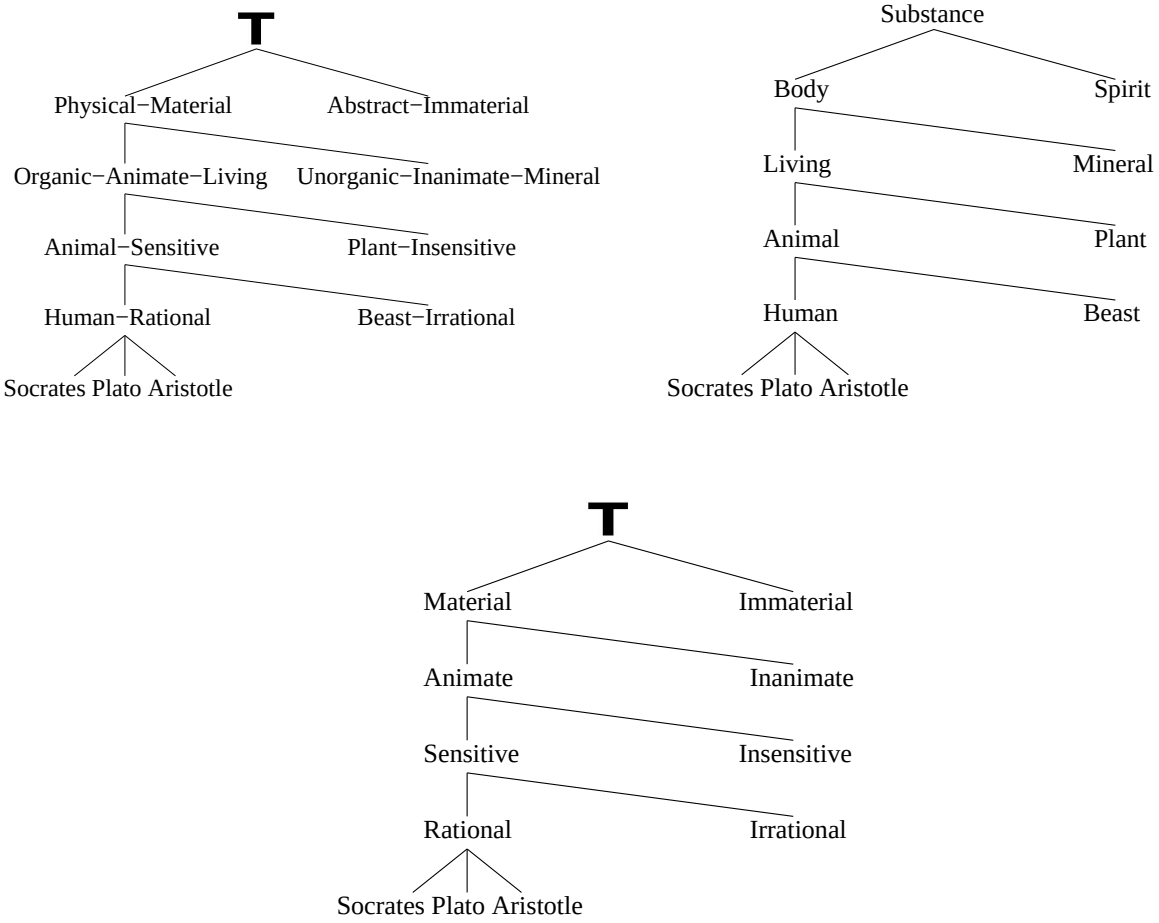


Figure 45: These three visualizations show that the same information that is in the Tree of Porphyry can be greatly simplified, yet not loosing any information.

Wilkins' Categories

The top level hierarchy of John Wilkins (1614-1672) is partly in resemblance with Aristotle's categories as figures 46 and 47 reveal. The example in figure 46 is taken from Wilkins' 376-page book about the English grammar: *An Essay Towards a Real Character and A Philosophical Language* [152].

The categories that are typed in capital letters and have a Roman index number in the end are pointers to other category trees in the book. The book contains a large ontology, i.e., similar kinds of categorizations than in figure 46 which are drawn from left to right instead of top to bottom because of representational reasons.

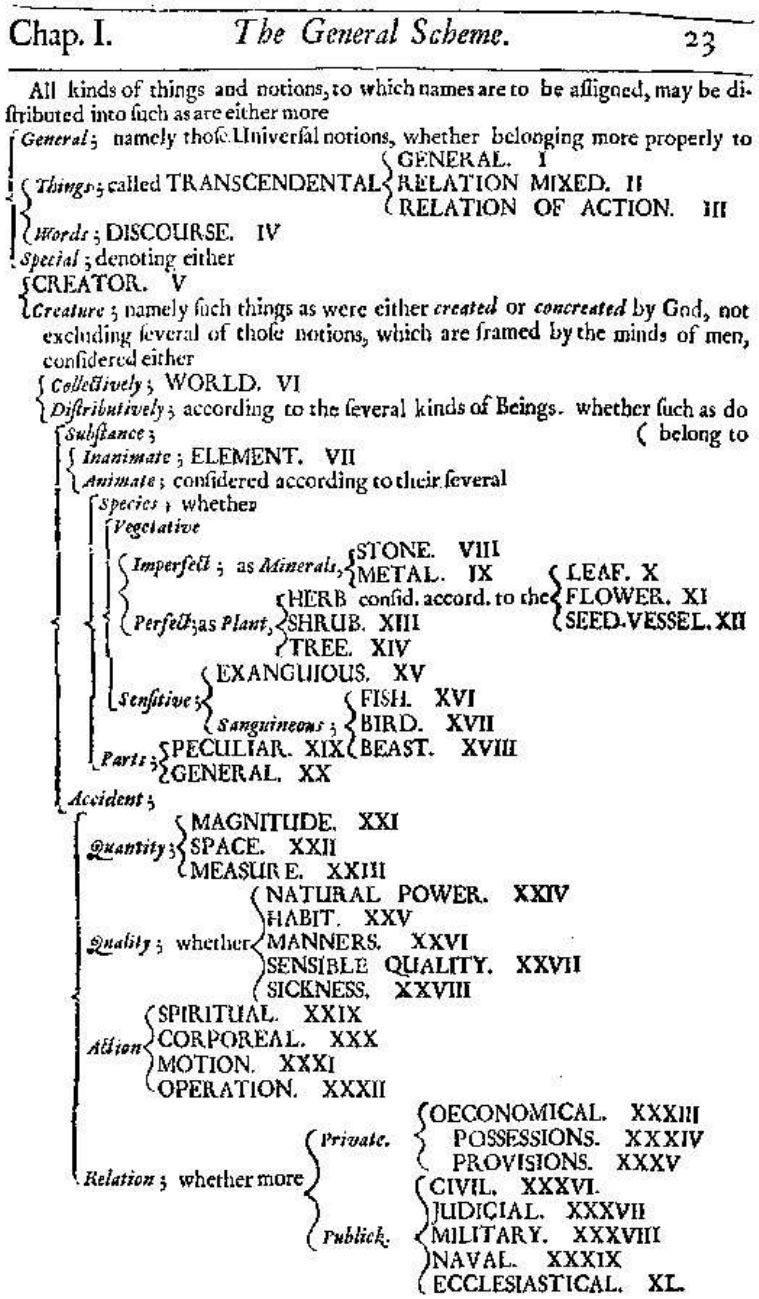


Figure 46: Top level hierarchy of John Wilkins.

The book is organized in four chapters: chapter 1 describes the history of language from the fall of Tower of Babel until 1700's, chapter 2 contains the ontology that defines the meaning of words, chapter 3 explains the English grammar, i.e., how the categories should be used in general, and chapter 4 gives examples of how the grammar is actually applied in creation of English sentences. Examples that relate hundreds of other human languages to the ontology are included.

There are many categories in Wilkins' top level hierarchy similar to those of Aristotle, but Wilkins has specified a greatly deeper hierarchy, and the meaning of Wilkins' categories is harder to understand than of those few categories in the Tree of Porphyry. Nevertheless, also Wilkins' layout follows the rules of category tree. Figure 47 clarifies some of Wilkins' top level hierarchy, but understanding the whole tree would require exploring the categories deeper in the hierarchy.

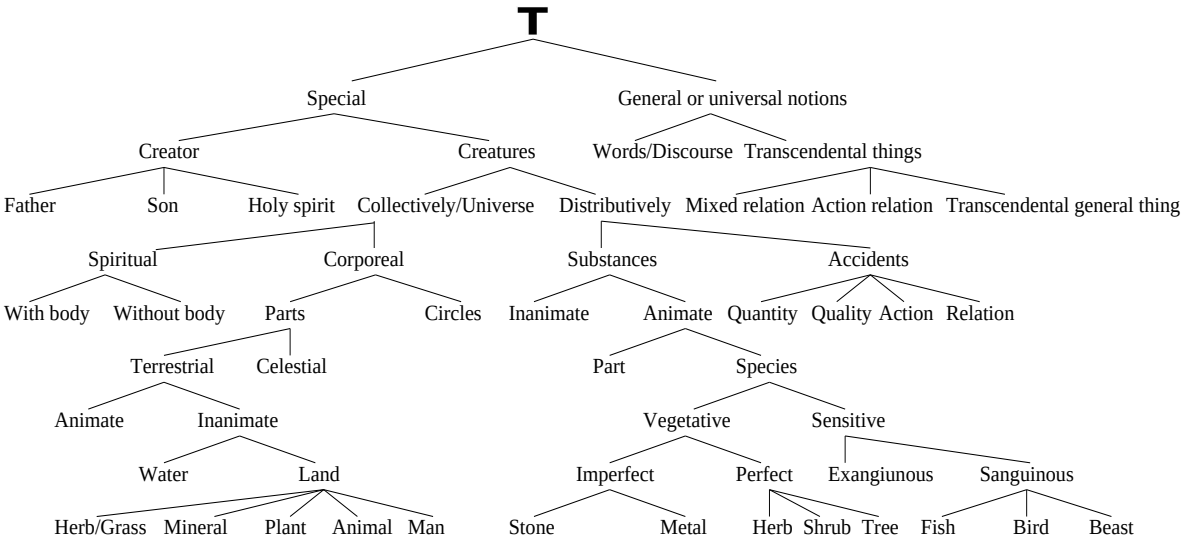


Figure 47: Clarification of a part of Wilkins'es top level hierarchy.

General or universal notions could be taken as **Abstract** of Plato's tree and **Immaterial** of the Tree of Porphyry, and **Special** as **Physical** of Plato's tree and **Material** of the Tree of Porphyry, but since **Accidents** is a successor of **Special**, and **Accident** clearly describes abstract things, this does not seem to be a clear case.

As can be seen, Wilkins 'used' the idea of multiple inheritance of categories: there are for example two paths of categories leading to classifications of animals and two paths leading to classification of plants, as cab be seen on the bottom level of figure 47. The classification of animals and plants are two of the deepest categorizations in Wilkins' book.

Brentano's Tree

The next example shows another visualization of Aristotle's categories. The tree in Figure 48 is based on a diagram by Franz Brentano (1838-1917), originated in 1862 [142]. In the first treatise of Aristotle's collected works [5], Aristotle presented ten basic categories, which are shown as the leaves of the tree in Figure 51. To connect the categories of Figure 48, Brentano added some terms taken from other works by Aristotle, including the top node **Being** and the terms at the branching categories: **Accident**, **Property**, **Inherence**, **Directedness**, **Containment**, **Movement**, and **Intermediacy**.

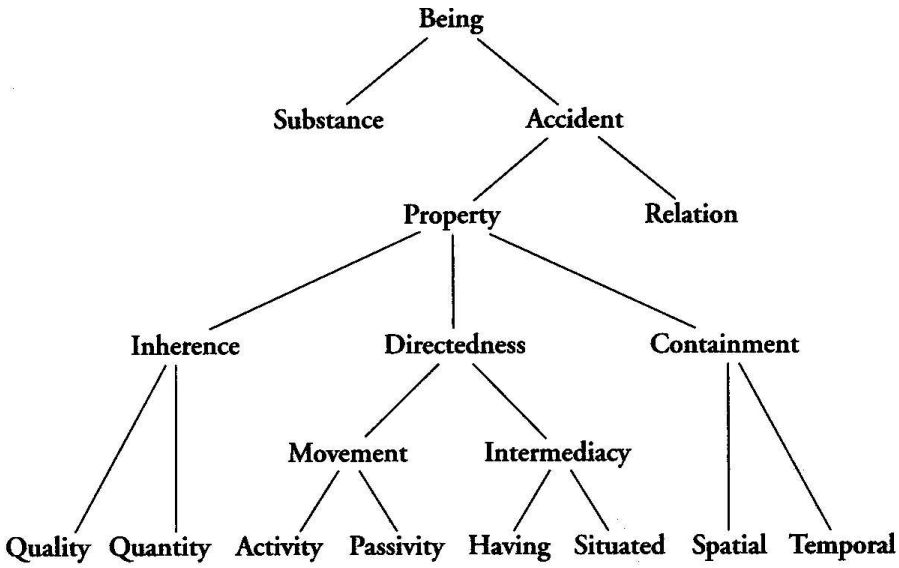


Figure 48: Brentano's tree of Aristotle's categories.

The top level categories of CYC [142, 27] in figure 49 are a clear example of how the meaning of categories comes from the meaning of their successors and predecessors, and how categories only describe things in contrast to each other.

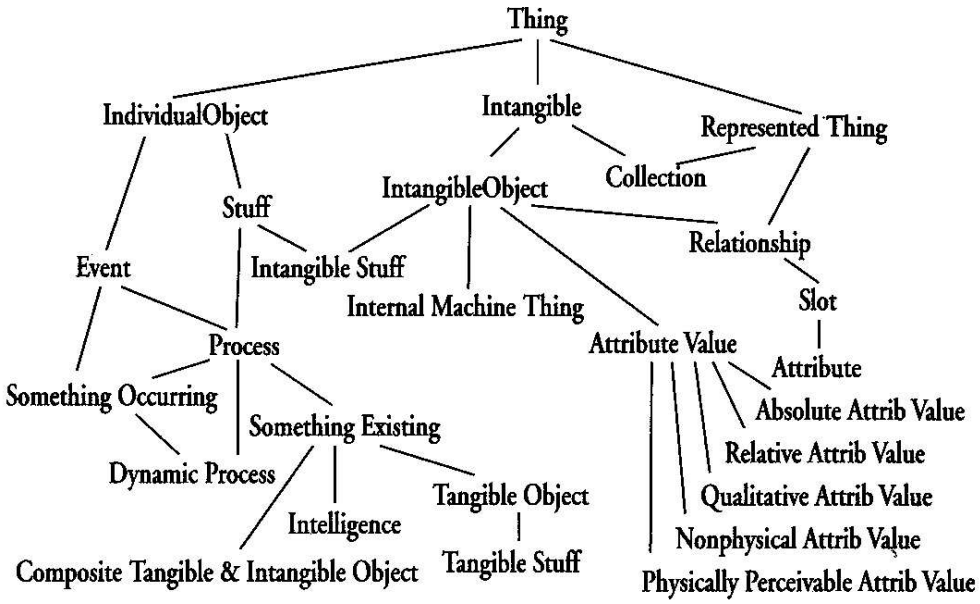


Figure 49: Top level categories of CYC.

CYC (enCYClopedia) [100, 60, 27] was initiated in the 1980's in course of providing common-sense reasoning support for Artificial Intelligence programs. Cyc provides the worldwide largest single formalized ontology, and can be seen as an extreme example of high level of formalization, when WordNet (p. 92) and ODP (p. 92) provide a low level of formalization. Hundreds of thousands of concepts have been formalized with millions of logical axioms, rules, and other assertions, which specify constraints of individual objects and classes. CycLanguage is used to express Cyc ontology. CycL's syntax is derived from first-order predicate calculus through the use of *second*-order concepts. Predicates express relationships of the categories and other entities of CycL. CycL has one universal quantifier *forAll*, and four existential quantifiers, *thereExists*, *thereExistAtLeast*, *thereExistAtMost*, and *thereExistExactly*. Additional quantifiers can be introduced by making the appropriate assertions. Logical connectives (*and*, *or*, and *not*) are used to build more complex formulas from other formulas. New connectives can be introduced simply by inserting a formula of that effect into the knowledge base. CycL accommodates a variety of truth-values (*default true*, *monotonically true*, *default false*, *monotonically false*, and *unknown*), fuzzy truth values, fuzzy attributes and sets, and Bayesian probabilities and dependencies.

CycL is a highly expressive language, but the pay-off with the expressive power is the usability and maintainability, which is the case also with early *Description Logic Language* knowledge bases [108, 31]. CycL has a different scope and purpose than the Semantic Web ontology languages for example, and is a lot of harder to take in use than these simple languages disregarding the intended usage, and the maintaining task of CYC is huge because of hundreds of thousands of axioms and complex rules. Description Logic Languages like Cyc certainly have a place in the toolkit of a conceptual modeler but they

have not gained much popularity as raw tools for conceptual modeling in the past [13]. The question remains: "Can Cyc give answers to the questions our applications need or do the distributed Semantic Web ontologies run over Cyc in the long run?"

Wordnet

WordNet was developed by the Cognitive Science Laboratory at Princeton University [25, 39, 157]. WordNet is an on-line lexical reference system where English nouns, verbs, and adjectives are organized into synonym sets, each representing one underlying lexical concept. The initial idea of WordNet was providing aid in browsing dictionaries conceptually, instead of trying to brows dictionaries alphabetically. Therefore, WordNet can be called a dictionary which is based on psycholinguistic principles.

WordNet divides the lexicon into five categories: *noun*, *verb*, *adjective*, *adverb*, and *function word*. WordNet organizes lexical information in terms of word meanings rather than word forms. Therefore, semantic relationships are used for organization. Examples of these relationships are synonym (similarity in meaning of words), antonym (dichotomy in meaning of words, for example, *victory* versus *defeat*), homonym (is-a relationship between words), meronym (part-of relationship between words), and morphological relations to reduce word forms. WordNet does not provide any definitions of semantics in a formal language, which leaves the definitions vague and limits the possibility for automatic reasoning support. At last count, the WordNet had grown into an unprecedented web of 138,838 English words linked in hundreds of thousands of different ways. A multilingual EuroWordNet [37] also exists in the meantime.

Open Directory Project

The *Open Directory Project ODP* [116] is the largest and most comprehensive human-edited directory on the Web. It is increasingly used for searching and annotating Web sites and is evolving all the time. ODP organizes Web sites in a category hierarchy. A user can annotate a Web page starting from ODP's top level hierarchy (fig. 50) that is used as a hyperlink structure: a user goes deeper in the hierarchy simply by clicking the currently presented categories. When the wanted category is found, the user can annotate a URI with a title and a description of the site. The user can naturally only search for something and not annotate anything. ODP is compared to text-based annotation and retrieval systems in section 5. Currently there are about 590,000 categories and 4 million site URI's annotated in the system. ODP supports multiple inheritance of categories, but does not support multi-instantiation, i.e., the ability to link a Web page to more than one category. This causes more necessary pay-offs in the category structure than with multi-instantiation. Currently a user can select only one category at a time, and other categories are revealed to the users as recommendations that are based on the successor-predecessor relationships of the categories.

ODP is hosted and administered by Netscape Communication Corporation, and is constructed and maintained by a vast, global community of over 61,000 volunteered editors, which facilitates the commitment of a large group to the ontology. ODP powers the core directory services for the Web's largest and most popular search engines including Netscape Search [111], AOL Search [4], Google [55], Lycos [106], HotBot [71], DirectHit [30], and hundreds of others.

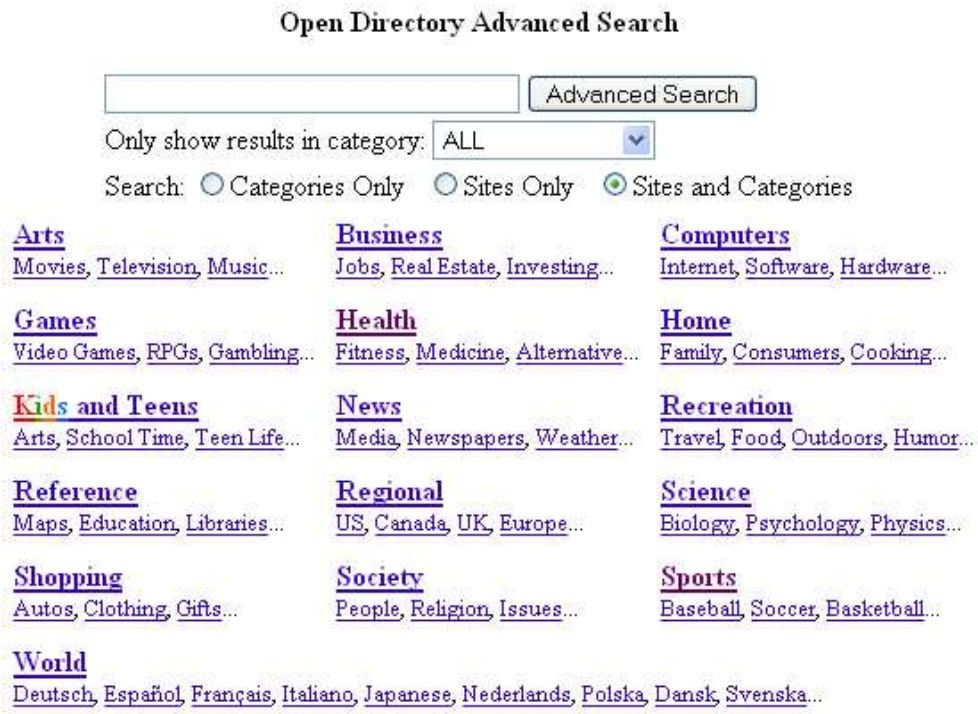


Figure 50: Top level categories of ODP.

Recently, there has appeared public complaints about ODP's editors subjectivity: if an editor has a commercial firm that has been placed under the same category that he/she edits, then the editor's personal interests endanger the submission of similar firms to that category. It can also take many months until the system properly accepts an annotation, and the user gets no notifications about how the process of acceptance proceeds.

Sowa's Diamond

The last and the newest well known example of a philosophical category tree is called *Sowa's Diamond*, created by John F. Sowa (1940-) [142, 143, 59].

Sowa's Diamond is very sophisticated and quite hard to understand at the first glance. It clearly integrates the insights of very many philosophers quite successfully, because it is not in contradiction with the other examples in this appendix, but only takes them further.

It is the only example of a category tree here where there are more subcategories of $\top$ than only **Physical** and **Abstract**. There are also **Continuant-Occurrent** and **Independent-Relative-Mediating**, which correspond to *Firstness, Secondness, Thirdness* (sect. 3.1.4) respectively. Even if these concepts are used *as* categories, the principle of triads is applied with all the categories, i.e, Firstness, Secondness, and Thirdness can be used also to explain the meaning of the categories.

There are not many categories that are *typed with the same character strings* than the English translations of Aristotle's categories, but many of Sowa's categories have a similar kind of meaning than Aristotle's categories. In addition to the correspondences that are clarified in the table in p. 85, the category that has been typed as **Property** in Brentano's tree, **Slot** and **Attribute** in CYC, and **Accidents** in Wilkins' tree, have been divided into three categories in Sowa's Diamond: **Actuality**, **Prehension**, and **Nexus**. Another similarity is that category **Abstract** has three subcategories in Sowa's Diamond: **Form**, **Proposition**, and **Intention**, which correspond to Plato's **Form**, **Hypothesis**, and **Idea** respectively.

The tree benefits also of multiple inheritance of categories as does Wilkins' tree. If the absurd type $\bot$ is disregarded, the categories on the lowest level can be taken as the leaves of the tree. The character strings that the titles of the leaves consist of could be changed, yet not changing the meaning of them: **Schema** could be turned into **Domain** or **Context**, **Description** into **Wiewpoint**, and **Reason** into **Goal** or **Objective**.

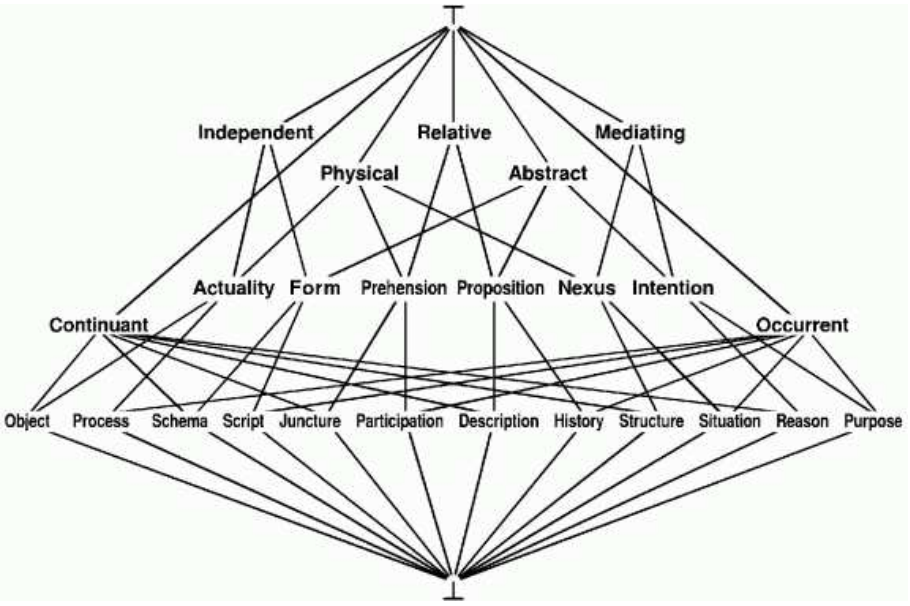


Figure 51: Sowa's Diamond