A Method for Modeling Uncertainty in Semantic Web Taxonomies

Markus Holi

Helsinki 23rd April 2004

Master of Science Thesis

UNIVERSITY OF HELSINKI
Department of Computer Science

Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Markus Holi

Työn nimi — Arbetets titel — Title

A Method for Modeling Uncertainty in Semantic Web Taxonomies

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master of Science Thesis	23rd April 2004	57 pages + 6 appendix pages

Tiivistelmä — Referat — Abstract

Semantic web ontologies are based on crisp logic, and do not provide well-defined means for expressing uncertainty needed when modeling the real world. To address this problem, this thesis presents a new probabilistic method to model degrees of subsumption, i.e., overlap between concepts. A notation is proposed, based on set theory, by which concepts can be quantified, and partial subsumption represented in a taxonomy. Based on this notation, a probabilistic method for computing degrees of overlap between the concepts is presented. Overlap is quantified by transforming the taxonomy into a Bayesian network. The degree of overlap is a simple, well-defined measure of conceptual similarity. It can be applied, for example, in information retrieval to computing the relevance relation of the result set.

ACM Computing Classification System (CCS):
I.2.3 [Deduction, Theorem Proving],
I.2.4 [Knowledge Representation Formalisms and Methods],
H.3.3 [Information Search and Retrieval]

Avainsanat — Nyckelord — Keywords
ontology, probabilistic reasoning, information retrieval

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

2 The Example Case RDF(S)

1 Introduction

Taxonomic concept hierarchies constitute an important part of the RDF(S) [BG] and OWL [SWM] ontologies used on the semantic web. For example, subsumption hierarchies based on the *subClassOf* or *partOf* properties are widely used.

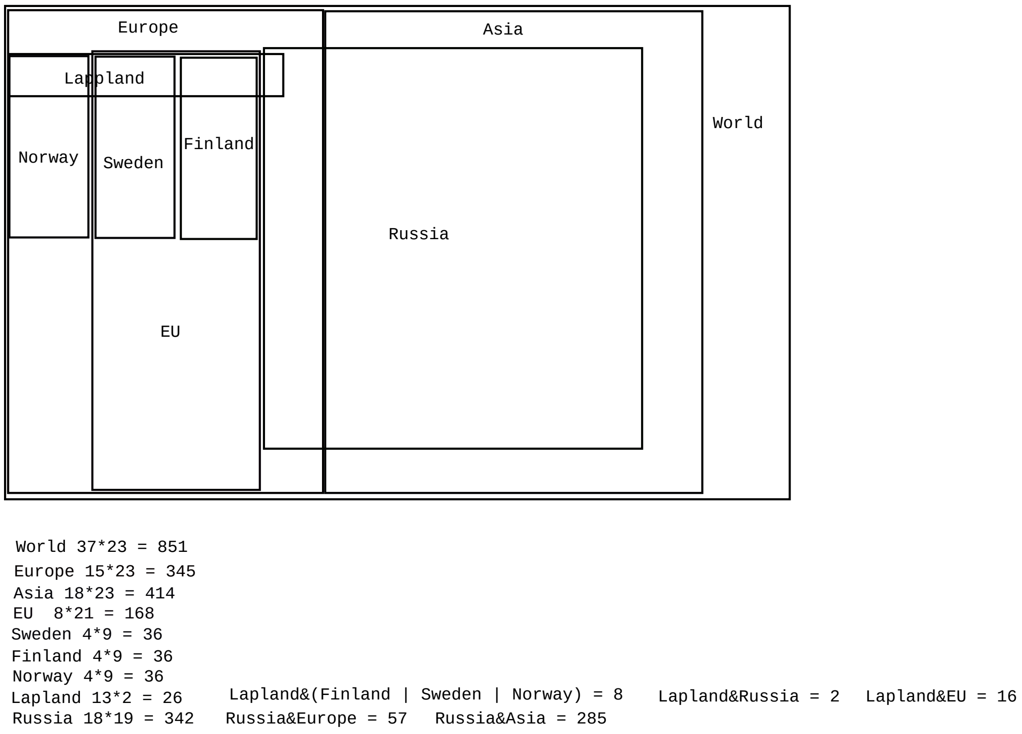

Figure 1: A Venn diagram illustrating countries, areas, their overlap, and size in the world.

Relations among real life entities are always a matter of degree. Subsumption hierarchies, on the other hand, are based on crisp logic. Thus, they fail to describe important aspects of real life concepts, and relations between them. This is an important drawback, that hinders the usability of ontology based information retrieval systems [WAS03].

The Venn diagram of figure 1 illustrates some countries and areas in the world. A crisp *partOf* meronymy cannot represent the partial overlap between the geographical area Lapland and the countries Finland, Sweden, Norway, and Russia, for example. A frequently used way to model the above situation would be to represent Lapland as the direct meronym of all the countries it overlaps, as in figure 2. This structure, however does not represent the situation of the map correctly, because

Lapland is not subsumed by anyone of these countries. In addition, the transitivity of the subsumption relation disappears in this structure. See, for example, the relationship between Lapland and Asia. In the Venn diagram they are disjoint, but according to the taxonomy, Lapland is subsumed by Asia.

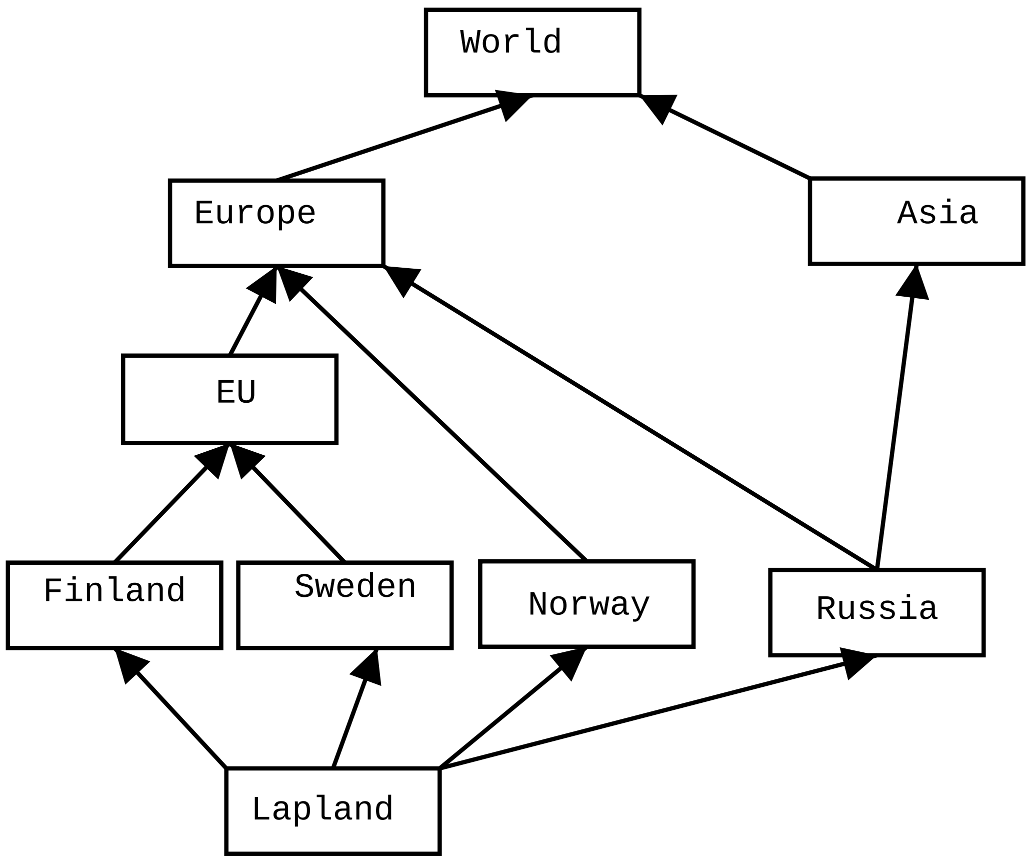

Figure 2: A standard semantic web taxonomy based on the Venn diagram of figure 1.

Another way would be to partition Lapland according to the countries it overlaps, as in figure 3. Every part is a direct meronym of both the respective country and Lapland. This structure is correct, in principle, but it too does not contain enough information to make inferences about the degrees of overlap between the areas. It does not say anything about the sizes of the different parts of Lapland, and how much they cover of the whole area of Lapland and the respective countries.

According to figure 1, the size of Lapland is 26 units, and the size of Finland is 36 units. The size of the overlapping area between Finland and Lapland is 8 units. Thus, 8/26 of Lapland belongs to Finland, and 8/36 of Finland belongs to Lapland. On the other hand, Lapland and Asia do not have any overlapping area, thus no part (0) of Laplan is part of Asia, and no part of Asia is part of Lapland. If we want a taxonomy to be an accurate representation of the 'map' of figure 1, there should be a way to make this kind of inferences based on the taxonomy.

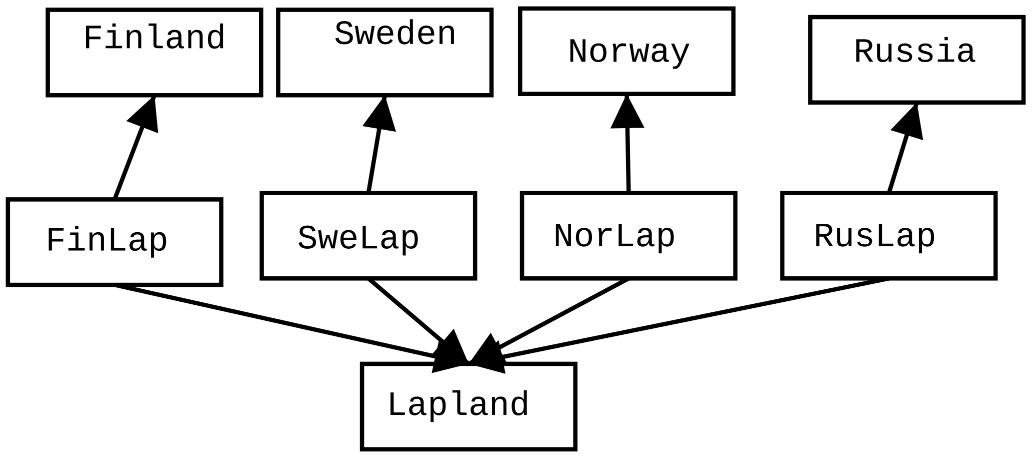

Figure 3: Representing Lapland's overlaps by partitioning it according to the areas it overlaps. Each part is subsumed by both Lapland and the respective country.

1.1 Problem Statement and Solution Approach

Based on the above it can be stated that with crisp taxonomies, it is not possible to quantify the coverage and the overlap between concepts. To address this foundational problem, this thesis presents a new probabilistic method to represent overlap in taxonomies, and to compute the overlap between a *selected* concept and every other, i.e. *referred* concept in the taxonomy. Thus, an *overlap table* is created for the selected concept. The overlap table can be created for every concept of a taxonomy. For example, table 1 present the overlap table of Lapland based on the the Venn diagram of figure 1. The Overlap column lists values expressing the mutual overlap of the selected concept and the other – referred – concepts, i.e. $Overlap = \frac{|Selected \cap Referred|}{|Referred|} \in [0, 1]$. These values can be used as natural measure of mutual overlap.

Intuitively, the overlap value has the following meaning: The value is 0 for disjoint concepts (e.g., Lapland and Asia) and 1, if the referred concept is subsumed by the selected one. High values lesser than one imply, that the meaning of the selected concept approaches the meaning of the referred one.

It is mathematically easy to compute the overlap tables, if a Venn diagram (the sets)

Selected	**Referred**	**Overlap**
Lapland	World	26/851 = 0.0306
	Europe	26/345 = 0.0754
	Asia	0/414 = 0.0
	EU	16/168 = 0.0953
	Norway	8/36 = 0.2222
	Sweden	8/36 = 0.2222
	Finland	8/36 = 0.2222
	Russia	2/342 = 0.0059

Table 1: The *overlap table* of Lapland according to figure 1.

is known. In practice, the Venn diagram may be difficult to create from the modeling view point, and computing with explicit sets is computationally complicated and inefficient. For these reasons the presented method calculates the overlap values from a taxonomic representation of the Venn diagram.

The presented method consists of two parts:

1. A graphical notation by which partial subsumption and concepts can be represented in a quantified form. The notation can be represented easily in RDF(S).
2. A method for computing degrees of overlap between the concepts of a taxonomy. Overlap is quantified by transforming the taxonomy first into a Bayesian network [FF01].

1.2 Potential Applications

Overlap values could be used in a number of ways. First, one problem in information retrieval systems is that, if the database is large, the query result sets are often very big. What is needed is a reasonable and effective way to rank the results. This can be done using the overlap values.

Assume that an ontology contains individual products manufactured in the different countries and areas of figure 1. The user is interested in finding objects manufactured in Lapland. The overlap values of table 1 then tell how well the annotations "Finland", "EU", "Asia", etc., match with the query concept "Lapland" in a well-defined probabilistic sense, and the hit list can be sorted into an order of relevance accordingly.

The overlap value between the selected concept (e.g. Lapland) and the referred concept (e.g. Finland) can in fact be written as the conditional probability $P(Finland'|Lapland')$ whose interpretation is the following: If a person is interested in data records about Lapland, what is the probability that the annotation "Finland" matches her query? X' is a binary random variable such that $X' = true$ means that the annotation "X" matches the query, and $X' = false$ means that "X" is not a match. This conditional probability interpretation of overlap values will be used in section 4.2 of this thesis.

A second application area could be ontology based recommendations. Because ontologies describe the semantic relations between the concepts of some domain, a reasoning system should, in principle, be able to guide the user through the data by recommendations based on the ontology. If, for example, we have a photograph repository annotated according to an ontology, and the user is looking at a photo taken in Finland, the system could guide the user to look also at photos of other countries, that are somehow similar or related, i.e. semantically close, to Finland. However, the notion of semantic closeness has proved to be a difficult concept to define and to use in the context of the ontologies of the semantic web. In some cases, for example in the Spectacle system [GC03], semantic closeness has been defined on the basis of the common instances of concepts. This seems intuitively plausible. However, it seems that this definition is too strict and only concepts that are very close to each other are touched by it. Based on the knowledge of the degree of overlap between concepts it is possible to create more interesting connections between concepts. For example, Finland and Sweden would not probably have any photographs annotated to both of them, however, they overlap with other concepts of the taxonomy in a similar way. This information could be used as an indication of semantic closeness between the two countries, and the ontology-based recommender could recommend photographs of Sweden to someone who is looking at a photograph taken from Finland.

Third, the knowledge of overlap between concepts could be used to find out how a concept has changed over time. For example, if we had a number of taxonomies representing the map of northern Europe at different times, we could make inferences, of how, for example, Finland and its relations to other countries have changed over time.

1.3 Chosen Method

The problem of representing uncertain or vague inclusion in ontologies and taxonomies has been tackled also by using methods of fuzzy logic [AWA+02, AP03, Zad65] and rough sets [SV00, Paw82]. See section 6 for a discussion of different methods. In this thesis, crisp set theory and Bayesian networks were used, because of the sound mathematical foundations they offer. The calculations are simple, but still enable the representation of overlap and uncertain subsumption between concepts. The Bayesian network representation of a taxonomy is useful not only for the matching problem discussed, but can also be used for other reasoning tasks, for example user modeling [KSO+01].

1.4 Organization of Thesis

The rest of the thesis is organized as follows. In chapter 2, the formalisms and methods used in this thesis are described. The discussed issues are the Semantic Web, RDF(S) ontologies, and Bayesian networks. In chapter 3, the graphical notation is defined. In chapter 4, the overlap calculation method is presented. In chapter 5, an implementation of the approach is described. In chapter 6, some related work is discussed. In chapter 7, the results of this thesis are summarized and discussed.

2 Background Knowledge

In this section some basic background knowledge to used formalisms and techniques will be given.

2.1 The Semantic Web

Most of the Web's content today is designed for humans to read, not for computer programs to manipulate meaningfully. Computers can parse Web pages for layout but in general, computers have no reliable way to process the semantics of the pages. The Semantic Web is an extension of the current Web in which information is given well-defined meaning, enabling computers to access the semantics of Web content. This facilitates the cooperation between computers and people. The Semantic Web is based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications [BLHL01]. One key idea of the Semantic Web is to associate semantically rich, descriptive information with any resource. For example, by adding meta-data about the creator of a document, we can search for documents created by different people.

On the Semantic Web not only documents are specified by URIs but more generally any resources, such as people, concepts, and even relationships between the different resources. Thus, the relationships between the different resources have a formal, machine-understandable specification. Moreover, by using URIs, things with the same name but different meaning can be differentiated. The vocabulary that is used in the description of resources can be specified in an ontology. Ontologies are an important part of the Semantic Web.

On the current Web, anyone can say anything about anything. The same will be true also about the Semantic Web. It is likely, that the Semantic Web will contain inaccurate and contradictory information. However, the software agents, functioning on the semantic web will have access to the ontologies, and to the reasoning mechanisms of each other. This way they can decide whether to trust answers given by other agents or not.

One important goal of the Semantic Web is also to support evolution of knowledge. The technologies are designed so that pieces of information can be easily combined, and changed if necessary.

2.2 Ontologies of the Semantic Web

An ontology is an explicit formal specification of the terms in the domain and relationships among them [Gru93]. Ontologies have been studied especially in artificial intelligence research. However, ontologies are becoming common in the WWW too.

The WWW Consortium (W3C) is developing the Resource Description Framework (RDF) to make the content of the web understandable to software agents searching for information. The RDF Schema (RDFS) is a basic framework to define machine understandable ontologies. Also other, more expressive ontology definition languages such as Web Ontology Language (OWL) are developed. Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their field. An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

There are a number of reasons for using ontologies [NM01].

1. To share common understanding of the structure of information among people or software agents. For instance, if two web sites share the same ontology, then each of them can benefit from the information of the other, e.g. in answering to user queries.

2. To enable reuse of domain knowledge. As a good example we can take the concept of time. Many different domains need to model time. If someone creates a good ontology to model time, then other domains can simply use it. Those previously developed ontologies can also be modified according to the needs of the re-users.

3. To make explicit domain assumptions. Often organizations have masses of implicit knowledge that is very hard to teach to novices. If domain assumptions are made explicit they can be learned easily, and also changed if they prove to be wrong.

4. To separate the domain knowledge from the operational knowledge. If we develop algorithms in which the domain knowledge is not hard-coded, but the knowledge comes from an ontology, then those algorithms may work in many different situations and domains.

2.2.1 The Elements of an Ontology

An ontology has four basic elements.

1. The concepts of the domain in question. The concepts are usually called the *classes* of the ontology.
2. Each concept has *properties* that describe various features and attributes of the concept.
3. There might be some *restrictions* on these properties, for example, what kind of values they can have.
4. The classes may have *instances.* Each instance represents an actual object that is a manifestation of the respective class.

Usually, concepts are the focus of the ontology. Concepts are represented by classes in the ontology. For example, the class *beer* represents all beers. Specific beers are instances of the class *beer.* A class can also have subclasses. For example, the class *beer* has the subclasses *ale* and *lager*, which in turn can have their own subclasses. Subclass hierarchies are called *taxonomies*, which are essential parts of ontologies. Properties describe various features of classes and instances. For example the property *yeast*, will have the value *top fermenting yeast* for the class *ale*, and *bottom fermenting yeast* for *lager.*

In practice, developing an ontology includes the following steps [NM01]:

1. Defining the concepts in the ontology. Usually they will be marked as classes.
2. Arranging the classes in a taxonomic (subclass-superclass) hierarchy.
3. Defining properties and describing allowed values for these slots.
4. Filling in the values for properties.

2.2.2 RDF and RDFS

RDF is a recommendation of W3C for representing meta-data about the resources of the web. RDFS provides facilities to define vocabularies by which resources can be described.

The basis of the RDF model is the triplet, which is constructed of a subject, a predicate, and an object. A resource (a subject) is linked to another resource (an object) through a directed arc labeled with a third resource (a predicate). In other words, a <subject> has a property <predicate> with value <object>. In RDF, all resources are identified by Unified Resource Identifiers (URI). URI provides a simple and extensible means for identifying a resource. The triplets can be presented as a directed graph for example, as in figure 4. The meaning of the triplet of figure 4 can be interpreted as 'MH is the creator of book.html'. An RDF triplet can also have a literal value as the object. A literal value is not a resource, but a string.

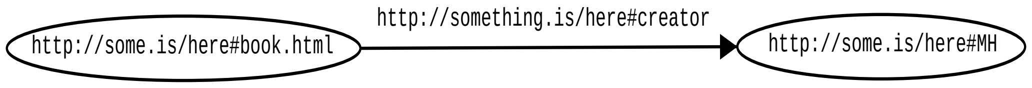
http://some.is/here#book.html
http://something.is/here#creator
http://some.is/here#MH

Figure 4: An RDF triplet.

A set of RDF triplets form a directed graph, whose nodes and arcs are labeled with URIs, as in figure 5. The graph of figure 5 states, that 'MH is the creator of both book.html and house.html, and MH lives in Finland'.

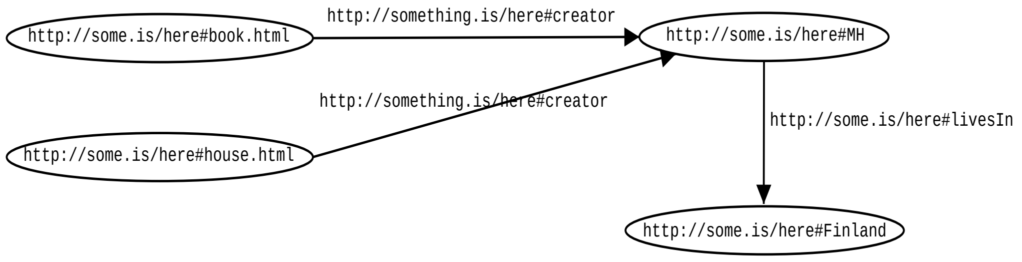
http://some.is/here#book.html
http://something.is/here#creator
http://some.is/here#MH
http://something.is/here#creator
http://some.is/here#house.html
http://some.is/here#livesIn
http://some.is/here#Finland

Figure 5: An RDF graph.

An RDF document is a list of descriptions. Each description applies usually to one resource, and contains a list of properties. Property values are either URIs, or literals. RDF documents can be encoded using the XML syntax. For example the triplet of figure 4 in XML as:

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<!DOCTYPE rdf:RDF [
    <!ENTITY rdf ``http://www.w3.org/1999/02/22-rdf-syntax-ns#''>
    <!ENTITY some  ``http://some.is/here#''>
]>
```

```
<rdf:RDF xmlns:rdf=''&rdf;''
         xmlns:some=''&some;''>
<rdf:Description  rdf:about=''&some;book.html''>
    <some:creator rdf:resource=''&some;MH''/>
</rdf:Description>
</rdf:RDF>
```

RDFS adds to the basic vocabulary of RDF some new primitives with which the creation of ontologies is possible. Such primitives are:

- *rdfs:Class* and *rdfs:subClassOf.* With these terms, classes and taxonomies of classes can be created.
- *rdfs:Resource.* This is the highest concept of any ontology. Every class in an ontology is the subclass of *rdfs:Resource.*
- *rdfs:domain* and *rdfs:range.* With these properties, restrictions on other properties can be expressed. *rdfs:domain* specifies the allowed classes, instances of which can be the subjects of the property. *rdfs:range* specifies the allowed value types of the property.
- *rdfs:subPropertyOf.* With this term taxonomies of properties can be created.
- *rdfs:Literal.* With this term, it can be expressed that a value of some property must be a literal.

Now, the beer ontology could be written with RDF(S) as follows:

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<!DOCTYPE rdf:RDF [
    <!ENTITY rdf ``http://www.w3.org/1999/02/22-rdf-syntax-ns#''>
    <!ENTITY some  ``http://some.is/here#''>
]>
<rdf:RDF xmlns:rdf=''&rdf;''
         xmlns:some=''&some;''>
<rdfs:Class rdf:about=''some;MetaClass''>
     <rdfs:subClassOf rdf:resource=''&rdfs;Class''/>
</rdfs:Class>
<some:MetaClass rdf:about=''&some;Beer''>
```

```
    <rdfs:subClassOf rdf:resource=''&rdfs;Resource''/>
</some:MetaClass>
<some:MetaClass rdf:about=''&some;Ale''>
    <rdfs:subClassOf rdf:resource=''&some;Beer''/>
    <some:yeast>top fermenting yeast</some:yeast>
</some:MetaClass>
<some:MetaClass rdf:about=''&some:Lager''>
    <rdfs:subClassOf rdf:resource=''&some;Beer''/>
    <some:yeast>bottom fermenting yeast</some:yeast>
</some:MetaClass>
<rdf:Property rdf:about=''some;yeast''>
    <rdfs:domain rdf:resource=''&some;Beer''/>
    <rdfs:range rdf:resource=''&rdfs;Literal''/>
</rdf:Property>
</rdf:RDF>
```

Usually, only instances of classes have properties, in an ontology. Because I wanted the classes to have properties - specifically the *some:yeast* property - I had to create a metaclass. The classes of the ontology are instances of this metaclass [BG] [LS] [Cha].

2.3 Bayesian Networks

Probabilistic reasoning is developed to deal with uncertainty. Agents almost never have access to the whole truth about their environment. For example, toothache usually indicates that the patient has a cavity. However, sometimes it indicates some other disease. The list of possible causes of toothache is almost unlimited.

Trying to use first-order logic to cope with a domain like medical diagnosis fails for three main reasons: First, it is too much work to list the complete set of facts that are somehow related to the domain. In fact, it can be argued that it is impossible. Second, we do not have complete theories about the world. Third, even if we had the theories, it would be untractably complex to try to find out all the needed facts about each practical case in order to make the right inferences. The agent's knowledge can at best provide only a degree of belief in the relevant sentences. The agent must, therefore, act under uncertainty. Probability theory deals with uncertainty, by assigning to each sentence a numerical degree of belief between 0 and 1.

With probability theory we can approximate, or summarize our knowledge about the domain and reason based on these approximations. A probabilistic agent has a belief state about the world. Beliefs about the world are updated based on perceptions. Based on these beliefs the agent chooses the action that yields the best result in the context of the agent's goals [RN03].

2.3.1 Basic Probability Theory

Degrees of belief are always assigned to propositions. The basic element of the language is the random variable, which can be thought of as referring to a part of the world whose status is initially unknown. Each variable has a domain of values such as *true* or *false*. The simplest kind of proposition asserts that a random variable has a particular value from its domain. The sum of the degrees of belief related to all the values of a variable is 1, i.e., $\sum_{a \in dom(A)} P(a) = 1$.

An atomic event is a complete specification of the state of the world about which the agent is uncertain. It can be thought of as an assignment of particular values to all the variables of which the world is composed. Atomic events are mutually exclusive — exactly one atomic event holds at any one point in time. Any particular atomic event entails the truth or falsehood of every proposition. Any proposition is logically equivalent to the disjunction of all atomic events in which a holds.

The unconditional or prior probability associated with a proposition a is the degree of belief accorded to it in the absence of any other information; it is written as $P(a)$. If we want to talk about the probabilities of all the possible values of a random variable the expression $P(A)$ is used, were A is the name of the variable in question. The sum of the probabilities for all the values of A is 1.

The joint probability distribution of a number of variables is the set of probabilities of all combinations of the values of the variables. If we include all the variables used to describe the world of an agent we get the *full joint probability distribution.* A full joint distribution specifies the probability of every atomic event and is therefore a complete specification of one's world.

When the agent obtains some knowledge about the world, the prior probabilities are no longer applicable. Instead, *conditional probabilities* are used. The notation is $P(a|b)$, which is interpreted as the degree of belief in the proposition a when all we know is b.

There is a connection between unconditional and conditional probabilities. The connection is expressed in the equation $P(a \wedge b) = P(a|b)P(b)$, which is called the *product rule.* It comes from the fact that, for a and b to be true, we need b to be true, and we also need a to be true given b.

Probability theory has three basic axioms:

1. For any a
   $0 \leq P(a) \leq 1$.
2. Necessarily true propositions have probability 1, and necessarily false propositions have probability 0.
3. The probability of disjunction is given by
   $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$.

Because any proposition a is equivalent to the disjunction of all the atomic events in which a holds, and atomic events are mutually exclusive, we can derive, with axiom 3, the following relationship: *The probability of a proposition is equal to the sum of probabilities of the atomic events in which it holds.* This relation provides a simple method for computing the probability of any proposition, given a full joint distribution that specifies the probabilities of all atomic events.

One common task in *probabilistic inference* is to find unconditional marginal probability distribution of a variable or a set of variables. If we know the *full joint probability distribution* we can do this by summing up all the atomic events in which the variable in question has the desired value. This procedure is called *marginalization.*

In most cases of *probabilistic inference* the interest in computing conditional probabilities of some variables, given evidence about others. Again, if we know the *full joint probability distribution* we can do this with the help of the *product rule.*

With the full joint distribution every task of *probabilistic inference* can be conducted. Nonetheless, this leads easily to intractably complex computations. Luckily, there are ways to reduce the complexity of the computations in most real-life situations.

A property that can in some cases reduce the complexity of probabilistic inference is *independence.* If propositions a and b are independent, then $P(a|b) = P(a)$, $P(b|a) = P(b)$, and $P(a \wedge b) = P(a)P(b)$. This means that in order to infer about two independent sets of variables, we can use two separate joint probability distributions, one for each set. This reduces the size of the distributions significantly. In many cases, however, these separate joint probability distributions too become too large for feasible computation. Moreover, it is often difficult to find completely independent sets of variables [RN03] [Jen96].

2.3.2 Bayesian Probability and Bayesian Networks

The equation that underlies all modern AI systems for probabilistic inference is

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}.$$

The equation is known as *Bayes' rule*, and it can be derived easily from the product rule. Bayes' rule does not seem very useful, because we need to know three probabilities just to calculate one. However, it is often the case, for example in diagnostic situations, that we have good estimates for the three. Probabilistic information is often available in the form of causal knowledge. For example, we know that if it rains, then the grass will most certainly get wet. However, if the grass is wet, what is the likelihood that it rained? This kind of knowledge is called diagnostic knowledge, and it is often more difficult to obtain, and more fragile, than causal knowledge.

There are cases in which we have more than one pieces of evidence, from which we want to infer the probability distribution of a variable. This leads quickly to similar complexity problems as those discussed in the previous section. Independence between variables reduces the complexity of probabilistic inference, but in many cases variables do not seem to bee independent of each other. However, often two or more variables are independent of each other given the presence or absence of another variable. This is called *conditional independence.* The situation in which variables A and B are independent given C, can be mathematically written as:

$$P(A \wedge B|C) = P(A|C)P(B|C) \tag{1}$$

Now $P(A|C)$ and $P(B|C)$ can be evaluated separately, which reduces the number of probabilities that need to be specified in order to define the full joint distribution. Thus, conditional independence assertions can allow probabilistic systems to scale up, and they are much more commonly available than absolute independence assertions.

A *Bayesian network* is a data structure to represent the dependencies among variables and to give a concise specification of any full joint probability distribution. A

Bayesian network is a directed graph in which each node is annotated with quantitative probability information. Each Bayesian network satisfies the following conditions:

1. A set of random variables makes up the nodes of the network. Variables may be discrete of continuous. In this thesis only discrete two state variables are considered.

2. A set of directed arcs connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y.

3. The graph has no directed cycles, and hence is a directed, acyclic graph (DAG).

4. Each node X is given a conditional probability distribution $P(X|Parents(X))$ that quantifies the effect of the parents on the node.

The topology of the network specifies the conditional independence relationships that hold in the domain. The intuitive meaning of an arrow in a properly constructed network is usually that X has a direct influence on Y.

In addition to the topology, a Bayesian network also includes *conditional probability tables* (CPT). A CPT is a table, in which each row contains the conditional probability of each node value for a conditioning case. A conditioning case is a possible combination of values for the parent nodes. Each row must sum up to 1.

A Bayesian network provides a complete description of the domain. Every entry in the full joint probability distribution can be calculated from the information in the network. A generic entry in the joint distribution is the probability of a conjunction of particular assignments to each variable, for example $P(x_1, \ldots x_n)$. The value of this entry is given by the formula

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)), \tag{2}$$

where $parents(X_i)$ denotes the specific values of the parents. Each conditional probability $P(x_i|parents(X_i)$ of the above formula is specified in the *conditional probability table* (CPT) of the variable X_i in the Bayesian network.

If two variables of a Bayesian network are in such a relation with each other that changing the state of one will not affect the other, then they are said to be *d-separated* of each other.

There are algorithms by which probabilistic inference can be made given a Bayesian network. Those algorithms will be used in this work, however, the description of these algorithms is beyond the scope of this thesis.

3 Representing Overlap

The term concept is defined as an "idea or thought that corresponds to some distinct entity or class of entities, or its essential features" [PH01]. In the context of the ontologies of the semantic web, a concept defines a set of individuals and the subsumption property (e.g. subClassOf) denotes a subset relationship between the respective sets [SWM].

A taxonomy is therefore a set of sets and can be represented, e.g., by a Venn diagram. To be able to compute the overlap between concepts in a taxonomy, the taxonomy should be an accurate representation of the set theoretic structure of the concepts.

3.1 The Overlap Graph

If A and B are sets, then A must be in one of the following relationships to B.

1. A is a subset of B, i.e. $A \subseteq B$.
2. A partially overlaps B, i.e. $\exists x, y : (x \in A \land x \in B) \land (y \in A \land y \notin B)$.
3. A is disjoint from B, i.e. $A \cap B = \emptyset$.

Based on these relations, a simple graph notation has been developed for representing uncertainty and overlap in a taxonomy as an acyclic *overlap graph* (OG). An OG is constructed from the following elements:

1. Nodes represent concepts, and a number called *mass* is attached to each node. The mass of concept A , is a measure of the size of the set corresponding to A, i.e. $m(A) = |s(A)|$, where $s(A)$ is the set corresponding to A.
2. A solid directed arc from concept A to B denotes crisp subsumption $s(A) \subseteq s(B)$.
3. A dashed arrow denotes disjointness $s(A) \cap s(B) = \emptyset$.
4. A dotted arrow represents quantified partial subsumption between concepts, which means that the concepts partially overlap in the Venn diagram. The amount of overlap is represented by the *partial overlap value* $p = \frac{|s(A) \cap s(B)|}{|s(A)|}$.

In addition to the quantities attached to the dotted arrows, also the other arrow types have implicit overlap values. The overlap value of a solid arc is 1 (crisp subsumption) and the value of a dashed arc is 0 (disjointness). The quantities of the arcs emerging from a concept must sum up to 1. This means that either only one solid arc can emerge from a node or several dotted arcs (partial overlap). In both cases, additional dashed arcs can be used (disjointness). Intuitively, the outgoing arcs constitute a quantified partition of the concept. Thus, the dotted arrows emerging from a concept must always point to concepts that are mutually disjoint with each other.

Notice, that if two concepts overlap, there must be a directed (solid or dotted) path between them. Thus, if there is not a directed path between concepts A and B, then they are necessarily disjoint from each other, and there is no need to use the dashed arrow to express disjointness. If A and B are connected by a directed solid path, then the concepts necessarily overlap, and the disjointness relation can not be used between them. If two concepts are connected with a directed path that includes dotted arrows, then it is possible that they do not overlap. If this is the case, then disjointness between the concepts must be expressed explicitly using the disjointness relation, otherwise it is assumed that the concepts do overlap.

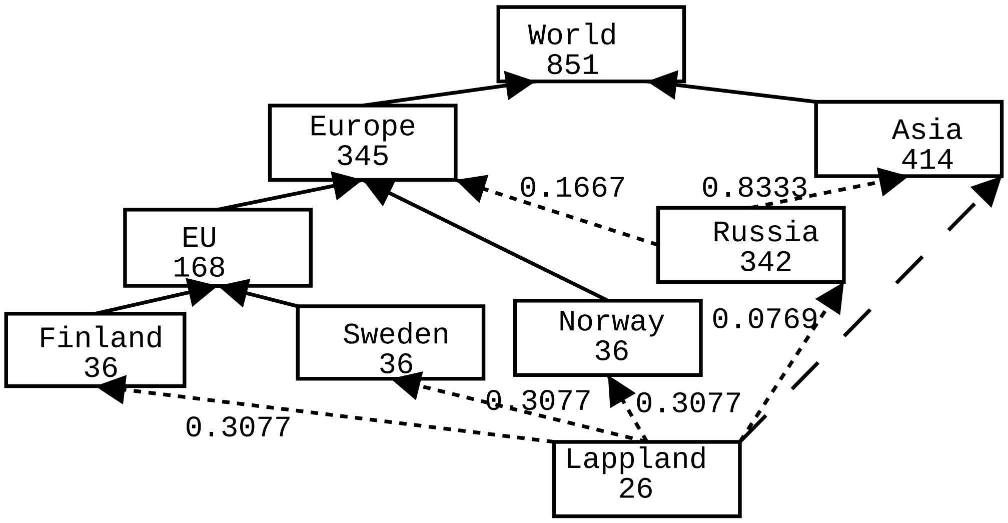

Figure 6: A taxonomy based on the Venn diagram of figure 1

For example, figure 6 depicts the meronymy of figure 1 as an OG. The geographic sizes of the areas are used as masses and the partial overlap values are determined based on the Venn diagram.

3.2 Completeness and Correctness of the Graphical Notation

In this section it is proved that any Venn diagram can be represented with the graph notation presented in the above section, and that this representation is correct. The motivation for this proof is that if any venn diagram can be represented with the graph notation, then also any collection of concepts can be represented as an overlap graph.

First, an algorithm for constructing an overlap graph based on a Venn diagram is presented, and then it is proved inductively that this algorithm yields an overlap graph which represents the Venn diagram correctly.

3.2.1 The Algorithm

The algorithm operates in an iterative manner. The main idea is that after every iteration, every distinct area in the part of the Venn diagram processed so far is represented by a node at the leaf layer of the overlap graph. Thus, the leaf layer of the overlap graph is a partition of the Venn diagram (the root set). The concepts added in each iteration are connected to the existing leaf layer by the means of the graph notation. This way the set theoretic structure of the overlap graph remains intact. Notice, that this algorithm requires the use of auxiliary concepts, which represent results of set operations over named sets, for example $s(A) \setminus s(B)$, where A and B are ordinary concepts. Algorithm 1 specifies the Venn to OG transformation.

In the first iteration the root set of the Venn diagram is added to the overlap graph. It will be the root node of the resulting overlap graph. After this, a set of mutually disjoint sets ($setsToBeAdded$) is extracted, in each iteration, from the Venn diagram. Each of these sets is added to the overlap graph and connected to the leaf layer of the existing overlap graph, according to the relations between the sets.

If A is a set to be added, and there exists a node B in the present leaf layer of the overlap graph such that $A \subseteq B$, then a solid arrow from A to B is added to the overlap graph.

If A partially overlaps sets $B_1 \ldots B_n$, then a dotted arrow is added from A to each of the overlapped sets. Because the idea of the algorithm is to represent each distinct area of the Venn diagram as a node in the overlap graph, nodes representing the intersections of A and each of $B_1 \ldots B_n$, need to be added to the overlap graph.

Each of these concepts $A \cap B_i$ is added as the complete subconcept of A and it is marked to be disjoint of all $B_1 \ldots B_n$ except B_i.

```
Data: VennDiagram V
Result: Taxonomy T
stepCounter = 0;
T = empty;
List namedSets = All the named sets in V;
Add the set that subsumes all other sets in V, to be the root in T;
while namedSets not empty do
    List setsToBeAdded = A list of mutually disjoint sets extracted from namedSets;
    List leafSets = the list of the sets added to T in the last iteration;
    foreach $s \in setsToBeAdded$ do
        Add s to T;
        if $\exists l \in leafSets : (s \subseteq l)$ then
            mark s as the complete subconcept of l in T ;
        end
        if s partially overlaps some leafSets in T then
            List overlappedSets = The list of overlapped leafSets;
            foreach $overlappedSet \in overlappedSets$ do
                mark s as the partial subconcept of overlappedSet;
                add $(s \cap overlappedSet)$ to T, as the complete subconcept of s;
                mark $(s \cap overlappedSet)$ to be disjoint from all other sets in overlappedSets;
            end
        end
    end
    foreach $leafSet \in leafSets$ do
        add a complement node to represent the part of leafSet that is not covered by any of the sets added above;
        mark it to be the complete subnode of leafSet;
    end
    ++stepCounter;
end
```

Algorithm 1: The algorithm by which any Venn diagram can be transformed into a taxonomy

After every set of *setsToBeAdded* is processed, additional nodes that represent those parts of concepts in the leaf layer of the last iteration that are not covered by *setsToBeAdded* are added to the overlap graph.

Notice that this algorithm is presented only to prove that it is possible to present any Venn diagram with the graph notation. It is not recommended that the overlap graphs would be created using this algorithm. Usually, it is not necessary to represent every distinct area explicitly in the overlap graph. For example, in figure 6, the part of Europe not covered by any of the countries in the Venn diagram of figure 1, is not represented explicitly, because the relations between the named sets in the diagram can be expressed without it.

3.2.2 The Proof

Theorem 1 *The algorithm adds every named set of the processed Venn diagram to the constructed overlap graph.*

PROOF:

In every iteration of the main loop (the While loop), at least one set is extracted from the list of named sets and added to the overlap graph. Thus, eventually all named sets get added.

Theorem 2 *Using algorithm 1, any Venn diagram can be transformed into an overlap graph. This overlap graph is an accurate representation of the Venn diagram.*

PROOF:

Because it is known, based on the above theorem, that every named concept is processed by the algorithm, it suffices to prove that after each step the overlap graph represents correctly every distinct area of the part of the Venn diagram processed so far. Each area is represented by a node in the overlap graph. Thus, after all sets are processed, the overlap graph is a representation of the whole Venn diagram.

The base case: Step 0.
The root concept is added. Now the overlap graph presents the part of the Venn diagram where only the root set is present. Thus, the processed part of the Venn diagram consists of one distinct area which is presented correctly by the root node.

Induction assumption: Step k.
It is assumed that the taxonomy represents correctly every distinct area of the part of the Venn diagram processed in the k steps of algorithm 1.

Induction step: Step $k+1$.
Based on the algorithm, it is known that the concepts to be added are mutually disjoint.
According to the induction assumption the sets corresponding to the concepts in the leaf layer of the present overlap graph are disjoint, and they form a partition of the Venn diagram.

Every distinct area of the corresponding part of the Venn diagram is represented correctly also after step $k+1$ because:

- Relations are made only to the leaf layer of the present overlap graph. Thus no cycles can be formed.

- Each concept is added to the overlap graph with the mass value that is derived from the Venn diagram (Size correct).
- Each concept will be connected only to the leaf concepts (concepts added at step k). Thus there will not be a connected path between any of the concepts added in this step (disjointness).
- Every relation is added based on the set theoretic structure of the Venn diagram. Thus, the individual relations between the sets processed in step $k+1$ and sets processed in step k are correct. Because the relations (arrows) in the graphical notation cover every possible relation between two sets, the overlap graph conforms to the graph notation, also after step $k+1$.
- Each distinct area inside a named set is also added to the taxonomy.
- After the named sets are added, each distinct area in the Venn diagram outside the named sets are also added.
- Because all distinct areas of the Venn diagram must be inside or outside the named concepts of step $k+1$, every distinct area is covered.

Thus, the algorithm yields an overlap graph conforming to the graph notation specified, and that overlap graph is an accurate representation of the set structure of the Venn diagram.

3.3 The Efficiency of the Graphical Notation

In the worst case, the size of the overlap graph grows exponentially with respect to the number of the named sets in the Venn diagram. Such a set structure can be seen in figure 7. In this case the described algorithm actually has to be used to construct the overlap graph. Only one named set per iteration is added, and each named set partially overlaps all the sets in the leaf layer of the existing overlap graph. In this case the number of the nodes $n(OG)$ in the resulting overlap graph can be calculated with the following formula.

$$n(OG) = n_0 + n_1 + \sum_{i=2}^{N-1} (2^i + 1) \approx 2^N, \qquad (3)$$

where n_i is the number of nodes added to the overlap graph in step i, $n_0 = 1$, $n_1 = 2$, and N is the number of named sets in the Venn diagram, including the root set.

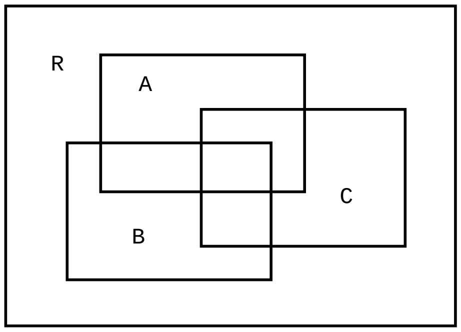

Figure 7: A Venn diagram which lead to intractable growth of the size of the resulting overlap graph.

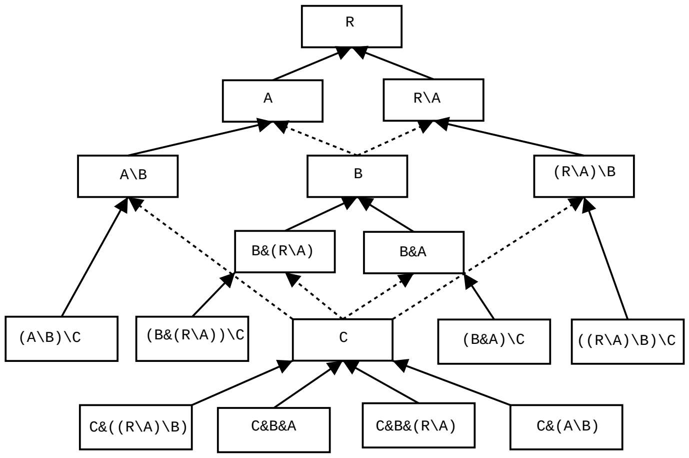

Figure 8: The overlap graph representing the Venn diagram of figure 7. Notice that the dashed arrows (disjointness), the masses of the concepts, and the quantities of the partial subsumptions are not shown in this figure, for the sake of simplicity.

The overlap graph representing the Venn diagram of figure 7 can be seen in figure 8. Notice that the arrows expressing disjointness are omitted from the figure, to keep the figure simple. In step 0, one set, $n_0 = 1$, is added, i.e. the root set. In step 1, the set A is processed, and nodes representing A and $R \setminus A$ are added to the OG. Thus, $n_1 = 2$. In step 2, B is processed. It partially overlaps both A and $R \setminus A$. According to the algorithm we must represent each distinct area of the part of Venn diagram that is represented so far by the OG. Thus, we must add four auxiliary concepts to the OG in addition to B, two of which represent areas inside B, and two outside B. Thus, $n_2 = 5 = 2^2 + 1$. When C is processed the number of auxiliary concepts is 8, and the total number of added nodes is then $9 = 2^3 + 1$.

Nonetheless, the above situation is theoretical. Overlap graphs are designed to represent overlap between real-world concepts. Based on experiences with concept modeling, it is argued in this thesis that such highly complex overlap structure never occur in practice, when modeling relationships between real-world concepts. In most cases, the number of nodes in the resulting overlap graph equals or nearly equals the number of named sets in the Venn diagram.

4 Computing the Overlap Values

The method presented in this thesis creates an overlap table (cf. figure 1) for each concept in the taxonomy. Computing the overlaps is easiest when there are only solid arcs, i.e., a complete subsumption relation between concepts. There are three ways in which two concepts A and B can be related, when the overlap graph contains only solid graphs between concepts.

1. There is a directed solid path from A (selected) to B (referred). In this case overlap $o = \frac{|s(A) \cap s(B)|}{|s(B)|} = \frac{m(A)}{m(B)}$, where $m(X)$ is the mass of concept X.
2. The solid path is directed from B to A. Now $o = \frac{|s(A) \cap s(B)|}{|s(B)|} = \frac{m(B)}{m(B)} = 1$.
3. There is not a directed path between A and B. Now $o = \frac{|s(A) \cap s(B)|}{|s(B)|} = \frac{|\emptyset|}{m(B)} = 0$.

If there is a mixed path of solid and dotted arcs between A and B, then the calculation is not as simple. Consider, for example, the relation between *Lapland* and *EU* in figure 6. To compute the overlap, we have to follow all the paths emerging from *Lapland*, take into account the disjoint relation between *Lapland* and *Asia*, and sum up the partial subsumption values somehow.

Because of the difficulties arising from the mixed paths, and on the other hand because of the straight forwardness of the solid path situation, the approach taken in this thesis is to transform the taxonomy into a structure containing only solid paths, and calculating the overlap tables of concepts based on this structure. The upper level plan for the quantification of overlap is specified in algorithm 2. Thus, in the transformed taxonomy structure the only relation between concepts will be the subsumption relation.

```
Data: Taxonomy T
Result: OverlapTables OT
SPS = TransformIntoSolidPathStructure(T);
OT = CreateOverlapTables(SPS);
```

Algorithm 2: The upper level algorithm for quantifying the overlap between concepts

4.1 Transformation into the Solid Path Structure

The transformation is done according to the following principle.

Transformation Principle 1 *Let A be the direct partial subconcept of B with overlap value o. In the solid path structure the partial subsumption is replaced by an additional* middle concept*, which represents $s(A) \cap s(B)$. It is marked to be the complete subconcept of both A and B, and its mass is $o \cdot m(A)$.*

See for example how the relationship between Lapland and Finland in figure 6 is transformed into the structure in figure 9.

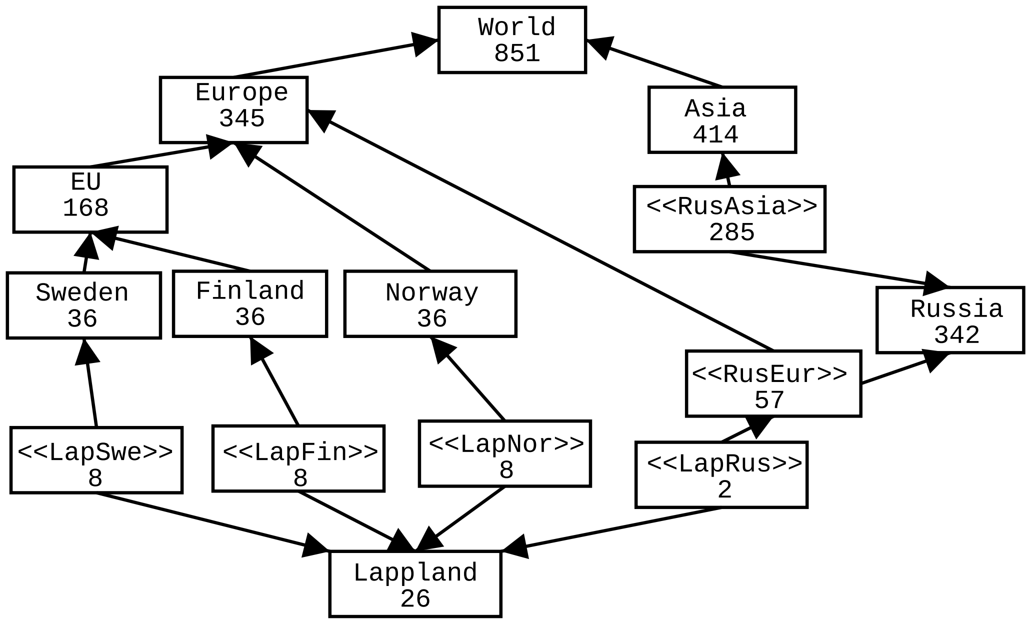

Figure 9: The taxonomy of figure 6 transformed into a *solid path structure* containing only subsumption relations

4.1.1 The Transformation Algorithm

The transformation is specified in algorithm 3. The algorithm processes the overlap graph T in a breadth-first manner starting from the root concept. A concept c is processed only after all of its super concepts (partial or complete) are processed. Because the graph is acyclic, all the concepts will eventually be processed.

Each processed concept c is written to the solid path structure SPS. Then each arrow emerging from c is processed in the following way. If the arrow is solid, indicating subsumption, then it is written into the solid path structure as such. If the arrow is dotted, indicating partial subsumption, then a middle concept $newMc$ is added into the solid path structure. It is marked to be the complete subconcept of both c and the concept p to which the dotted arrow points in T. The mass of $newMc$ is $m(newMc) = |s(c) \cap s(p)| = o \cdot m(c)$, where o is the overlap value attached to the dotted arrow.

```
Data: OverlapGraph T
Result: SolidPathStructure SPS
SPS := empty;
foreach concept c in T do
    foreach complete or partial direct superconcept p of c in T do
        if p connected to its superconcepts through middle concepts in SPS then
            mc := the middle concept that c overlaps;
            if c complete subconcept of p then
                mark c to be complete subconcept of mc in SPS;
            else
                newMc := middle concept representing
                $s(c) \cap s(p)$;
                mark newMC to be complete subconcept of c and mc in SPS;
            end
        else
            if c complete subconcept of p then
                mark c as complete subconcept of p in SPS;
            else
                newMc .= middle concept representing
                $s(c) \cap s(p)$;
                mark newMc to be complete subconcept of c and p in SPS;
            end
        end
    end
end
```

Algorithm 3: Creating the solid path structure

However, if p is connected to its superconcepts (partial or complete) with a middle concept structure in SPS, then the processing is not as simple. In that case, c has to be connected to one of those middle concepts. The right middle concept is found by using the information conveyed in the dashed arcs emerging from c. The right middle concept mc is the one that is not subsumed by a concept that is marked to be disjoint from c in the overlap graph. This is the middle concept that c overlaps. Notice, that if the overlap graph is an accurate representation of the underlying Venn diagram, then mc is the only middle concept that fulfils the condition. If c is a complete subconcept of p in the overlap graph T, then c is marked to be the complete subconcept of mc in SPS. If c is a partial subconcept of p in T, then it is connected to mc with a middle concept structure.

Notice that if c was connected directly to p, instead of mc, then the information conveyed in the dashed arrows, indicating disjointness between concepts, would have been lost. For example, if in figure 9 *Lapland* was connected directly to *Russia*, then the information about the disjointness of *Lapland* and *Asia* would be lost.

The running time of this algorithm is linear to the size of the OG, i.e. $O(N)$, where N is the number of arcs in the OG. Each arc is processed once. The processing of each arc takes 2 processing steps at maximum.

4.1.2 Transformation of Sloppy Overlap Graphs

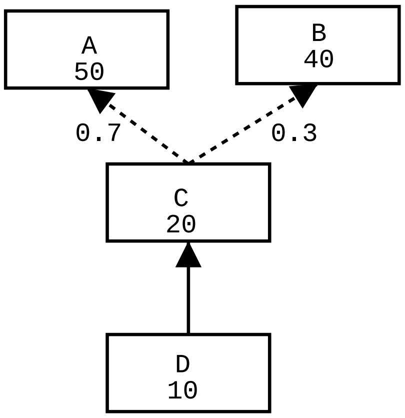

Figure 10: Sloppiness in the overlap graph.

The algorithm presented above is designed to work on an overlap graph that is an accurate representation of a Venn diagram. For the sake of usability, the graphical notation is designed so that an overlap graph may also contain some sloppiness. For example, see figure 10. In the figure concept D partly overlaps both A and B, and the quantities of the overlaps are not known. The transformed structure is presented in figure 11. D is connected with a middle concept structure to each of the middle classes connecting C to A and B. The mass is given based on the quantities of the partial subsumption relations between C and its superconcepts.

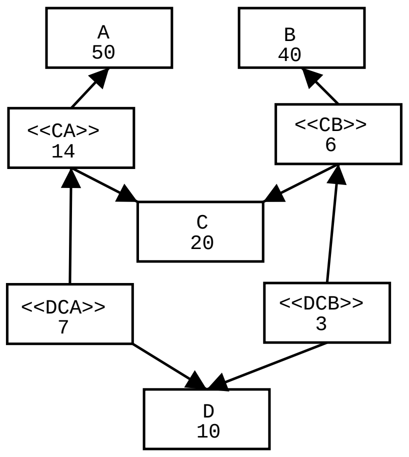

Figure 11: The sloppy overlap graph of figure 10 transformed into a solid path structure.

4.2 Using a Bayesian Network

Based on the solid path structure, the overlap table values o for a selected concept A and a referred concept B could be calculated by the algorithm 4, where notation X_s denotes the set of (sub)concepts subsumed by the concept X.

```
if $A$ subsumes $B$ then
    $o := 1$
else
    $C = A_s \cap B_s$
    if $C = \emptyset$ then
        $o := 0$
    else
        $o := \frac{\sum_{c \in C} m(c)}{m(B)}$
    end
end
```

Algorithm 4: Computing the overlap

The evaluation of overlap between Europe (selected) and Lapland (referred), for example, is done in the following way. First, all the concepts subsumed by Europe are selected, because they are constituents of Europe. Second, the masses of all those selected concepts that are also subsumed by Lapland are summed together. In effect, this is the mass of the union of the sets corresponding to the concepts , in this case

$$|s(LapSwe) \cup s(LapFin) \cup s(LapNor) \cup s(LapRus)|.$$

Third, the overlap value between Europe and Lapland is then

$$\frac{m(LapSwe) + m(LapFin) + m(LapNor) + m(LapRus)}{m(Lapland)} = 1.$$

This can be verified by the Venn diagram of figure 1.

The overlap table for A could be implemented by going through all the concepts of the graph and calculating the overlap value according to the above algorithm. However, because the overlap values between concepts can be interpreted as conditional probabilities (see page 5 of section 1), the solid path structure was chosen to be used as a Bayesian network topology. In the Bayesian network the boolean random variable X' replaces the concept X of the solid path structure. The efficient evidence propagation algorithms developed for Bayesian networks [FF01] take care of the overlap computations. Furthermore, a Bayesian representation of the taxonomy could have also some other uses. It could be used, for example, in user modeling [KSO$^+$01].

Recall from page 5 that if A is the selected concept and B is the referred one, then the overlap value o can be interpreted as the conditional probability

$$P(B' = true | A' = true) = \frac{|s(A) \cap s(B)|}{|s(B)|} = o \tag{4}$$

where $s(A)$ and $s(B)$ are the sets corresponding to the concepts A and B. A' and B' are boolean random variables such that the value *true* means that the corresponding concept is a match to the query, i.e, the concept in question is of interest to the user. $P(B'|A')$ tells what is the probability that concept B matches the query if we know that A is a match. The Venn diagram from which $s(A)$ and $s(B)$ are taken is not interpreted as a probability space, and the elements of the sets are not interpreted as elementary outcomes of some random phenomenon. The overlap value between $s(A)$ and $s(B)$ is used merely as a means for determining the conditional probability defined above.

The joint probability distribution of the Bayesian network is defined by marginal conditional probability tables (CPT) $P(A'|B'_1, B'_2, \dots B'_n)$ for nodes with parents $B'_i, i = 1 \dots n$, and by prior marginal probabilities set for nodes without parents. The CPT $P(A'|B'_1, B'_2, \dots B'_n)$ for a node A' can be constructed by enumerating the value combinations (true/false) of the parents $B'_i, i = 1 \dots n$, and by assigning:

$$P(A' = true|B'_1 = b_1, \ldots B'_n = b_n) = \frac{\sum\limits_{i \in \{i: b_i = true\}} m(B_i)}{m(A)} \tag{5}$$

where $m(X)$ is the mass of concept X. Recall that $m(X) = |s(X)|$. The value for the complementary case $P(A' = false|B'_1 = b_1, \ldots B'_n = b_n)$ is obtained simply by subtracting from 1. The above formula is based on the above definition of conditional probability, and algorithm 4. The intuition behind the formula is the following. If a user is interested in Sweden and in Finland, then she is interested both in data records about Finland and in data records about Sweden. The set corresponding to this is $s(Finland) \cup s(Sweden)$. In terms of the OG this is written as $m(Finland) + m(Sweden)$. In the Bayesian network both $Finland'$ and $Sweden'$ will be set "true". Thus, the bigger the number of European countries that the user is interested in, the bigger the probability that the annotation "Europe" matches her query, i.e., $P(Europe'|Sweden', Finland') > P(Europe'|Finland')$.

If A' has no parents, then $P(A' = true) = \lambda$, where λ is a very small non-zero probability, because the posterior probabilities are wanted to result from conditional probabilities only, i.e., from the overlap information.

The whole overlap table of a concept can now be determined efficiently by using the Bayesian network with its conditional and prior probabilities. By instantiating the nodes corresponding to the selected concept and the concepts subsumed by it as evidence (their values are set "true"), the propagation algorithm returns the overlap values as posterior probabilities of nodes.

Notice that when using the Bayesian network in the above way, a small inaccuracy is attached to each value as the result of the λ prior probability that was given to the parentless variables. This error approaches zero as λ approaches zero. The Bayesian network was defined in the above manner for the following reasons.

1. To be able to easily use the the solid path structure as the topology of the Bayesian network. The CPTs can be calculated directly based on the masses of the concepts.

2. With this definition the Bayesian evidence propagation algorithm returns the overlap values readily as posterior probabilities. Experiments were conducted with various ways to construct a Bayesian network according to probabilistic interpretations of the Venn diagram. Nonetheless, no one of these Bayesian networks answered to the basic need of sorting hits in an information retrieval

system in such a straight forward manner as the construction described in this thesis.

3. In the Bayesian network structure of this thesis d-separation indicates disjointness between the corresponding concepts. I see this as a useful characteristic, because it makes the simultaneous selection of two or more disjoint concepts possible.

5 Implementation

The presented approach was implemented as a proof-of-concept. The implementation includes both the overlap graph of section 3, and the overlap quantification algorithms of section 4.

5.1 Overlap Graph

The overlap graphs are implemented using RDF(S) in the following way. The full RDF(S) specification is presented in Appendix 1.

Concepts. In this implementation concepts can be represented either as RDF(S) classes or instances. For example the geographical overlap graph of figure 6 is implemented as a taxonomy of RDF(S) instances of the class *Place*. The masses are expressed using a special *Mass* class, which has two properties: subject and mass. Subject points to the concept in question, and mass tells the mass of the concept. Thus, for each instance an instance of the *Mass* class is created.

Subsumption. The subsumption element can be implemented with a property of the user's choice.

Partial subsumption. Partial subsumption is implemented by creating a special *PartialSubsumption* class, which has three properties: subject, object and overlap. The subject property points to the direct partial subconcept, the object to the direct partial superconcept, and overlap tells the amount of overlap between the two. Each partial subsumption relation is implemented by instantiating the *PartialSubsumption* class.

Disjointness. The disjointness element is implemented by *disjointFrom* property. A similar property is already an element of the OWL language.

5.2 Implementation of Overlap Quantification

The architecture of the application implementing the overlap quantification approach presented in section 4 is presented in figure 12.

As an input, the implementation takes an RDF(S) ontology, the URI of the root node from which the inclusion relations are considered, and the URI of the subsumption

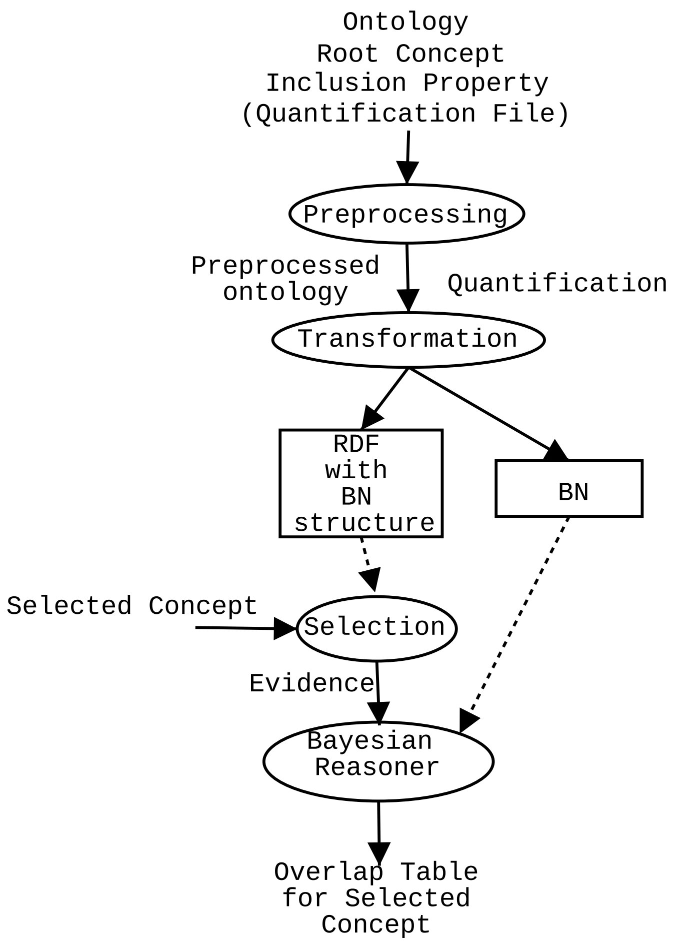

Figure 12: The architecture of algorithm implementation

property used in the ontology. Additionally, also an RDF data file that contains data records annotated according to the ontology may be given. The RDF(S) ontology must include a taxonomy conforming to the implementation of the graphical notation presented above. The root node represents the concepts that subsumes all other concepts in the taxonomy. Because the implementation of the graphical notation did not restrict the properties that may implement the subsumption property, it must be given as an input.

As an output the application gives the overlap tables for every concept in the taxonomy extracted from the input RDF(S) ontology.

The *preprocessing*, *transformation*, and selection modules are implemented with SWI-Prolog[1]. The Semantic Web package is used. The *Bayesian reasoner* mod-

[1]http://www.swi-prolog.org/

ule is implemented in Java, and it uses the Hugin Lite 6.3[2] through its Java API. The main program is written in Perl.

Next, each module is discussed separately.

5.2.1 The Main Program

The main program functions as a glue between the different modules of the application. It operates each of the modules according to the architecture in figure 12.

The main program contains a loop that gives each concept of the taxonomy to the selection module and then runs the Bayesian reasoner module with the selection. Thus, the application produces the overlap tables of all the concepts in the input taxonomy.

5.2.2 Preprocessing

Because different users may use different subsumption properties, and because the concepts can be RDF(S) classes or instances, they must be transformed into a predefined standard form. In the standard form all the concepts are RDF(S) classes, and the subsumption relation is the *subClassOf* property.

All the elements of the ontology except the subsumption predicates, the *PartialSubsumption* instances, the *mass* instances, and the RDF(S) classes and instances that are connected to the root concept through subsumption or partial subsumption, are filtered out of the ontology. Thus, the only thing that is left of the input RDF(S) ontology for further processing is the taxonomy.

Additionally, if also an RDF data file that contains data records annotated according to the ontology is given as input, then the preprocessing module creates the masses of the concepts of the taxonomy based on these annotations. It is done in the following way. First, for each concept, a sum of all the data records annotated directly to the concept is counted. Because the concepts, or part of the concepts that are subsumed by it, constitute a part of that concept, their masses have to be added to the sum of annotations. In the case of partial subsumption, a factor of the partially included concept is added. This quantification method is illustrated in figure 13.

[2]http://www.hugin.com/

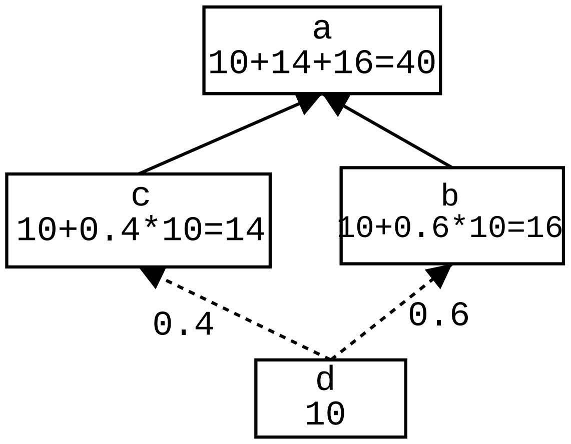

Figure 13: Quantification of concepts. The number of direct annotations to each concept is 10.

5.2.3 Transformation

The *transformation* module takes the taxonomy that was extracted from the ontology by the preprocessing module and transforms it into a Bayesian Network. Thus, this module implements algorithm 3 and creates the CPTs.

In addition to the Bayesian network, also an RDF(S) graph with an identical topology to the BN is created. These graph is used by the selection module while expanding the selection to include also the concepts subsumed by the selected concept. The Bayesian network is represented in the *.net* structure of Hugin.

5.2.4 Selection

The *selection* module takes an URI of the selected concept as an input and expands the selection to include all concepts that are subsumed by the selected one in the solid path structure. The *selection* module gives a list of the selected concepts as output.

5.2.5 Bayesian Reasoner

The Bayesian reasoner takes the Bayesian network and the list of selected nodes as input and mediates them to the Hugin decision engine, which returns the posterior probabilities to the module. The module writes these probabilities into the overlap table of the selected concepts.

5.3 Example Case

As an example case, the small geographical ontology of figure 6 is used. In the RDF(S) implementation, the places are instances of the class *Place*. The *partOf* relation is used as the subsumption property. The RDF(S) file can be seen in appendix 2. The taxonomy is created based on the Venn diagram of figure 1.

Sweden'	true		false	
Finland'	true	false	true	false
true (EU')	0.4285	0.2143	0.2143	0.0
false (EU')	0.5175	0.7857	0.7857	1.0

Table 2: The conditional probability table for EU' in the created Bayesian network

Selected	**Referred**	**Value**	**Expected Value**
Lapland	World	0.0306	26/851 = 0.0306
	Europe	0.0754	26/345 = 0.0754
	Asia	0.0	0/414 = 0
	EU	0.0953	16/168 = 0.0952
	Norway	0.2222	8/36 = 0.2222
	Sweden	0.2222	8/36 = 0.2222
	Finland	0.2222	8/36 = 0.2222
	Russia	0.0059	2/342 = 0.0058

Table 3: The *overlap table* of Lapland as calculated by the Bayesian network. The *Expected Value* column lists the overlap values that are calculated directly from the Venn diagram.

As the taxonomy was created based on the Venn diagram of figure 1, the Venn diagram can be used as an objective indicator of the accuracy of the implementation. By looking at the Venn diagram, the overlap values between concepts can be determined, and they can be compared to the values given by the implementation.

The solid path structure that was created based on the taxonomy, can be seen in figure 9. It is also the topology of the Bayesian network. An example of a CPT can be seen in table 2.

Table 3 presents the overlap table of Lapland as calculated by the Bayesian network, and as calculated from the Venn diagram. The slight differences in the overlap values of EU and Russia are results from the fact that in the taxonomy the overlap values are given with the precision of 4 significant digits, and also because of the small λ value given to the nodes without parents.

6 Related Work

In this section, alternative approaches to model uncertain and vague knowledge are described and discussed from the view-point of representing uncertainty in taxonomies.

6.1 Fuzzy Logic

In order to represent vagueness and uncertainty, Lotfi Zadeh [Zad65] developed fuzzy logic. Fuzzy logic is a logic with a continuous range of possibilities from 0 for impossible and 1.0 for certain. Fuzzy logic is based on fuzzy set theory introduced by Zadeh in 1965 [Zad65]. In fuzzy set theory there is a continuous range of membership values for the inclusion operator. Zadeh also assigned numeric values to hedging terms like almost, more or less, likely or very likely.

Fuzzy logic represents intuitive or subjective judgments. By doing this, it can sometimes handle situations with vague data. It would be very difficult to create statistical tables that relate person's age to the probability of being called young or old, for example. With subjective assignments of fuzzy logic this can be done easily. Zadeh proposed functions to represent different qualifiers. To represent the qualifier very, for example he proposed to square the values of the base distributions, e.g. $VeryYoung(x) = (young(x))^2$.

The most successful applications of fuzzy techniques have been in control systems for camera focus, elevator scheduling, subway brakes etc. Fuzzy control systems do not behave like theorem provers, but more like analog computers that simulate continuous phenomena [Men00].

6.1.1 Criticism of Fuzzy Logic

Fuzzy logic assigns all-purpose certainty factors to statements. However, usually certainties and statements are context-dependent. For example, a 19 years old person would be considered young as a university student but old as a high-school student. Human experts reason differently in different context and seem to have implicit understanding of each context. In the absence of information about context, fuzzy logic and related statistical methods may be useful as a first approximation.

A central goal of fuzzy logic is to make formal systems act more like human beings.

However, according to Susan Haack [Haa96], fuzzy logic models the way people talk and think very poorly. For example, in fuzzy logic we might assign a certainty factor of 0.8 to the proposition *Birds fly.* However it doesn't say anything about the particular cases. Penguings don't fly for certain, and healthy sparrows fly for certain. According to Susan Haack people do not reason based on likelihoods but according to context-dependent conditions.

Fuzzy logic helps to assign certainty factors according to various situations. However, it does not make the reasoning in any particular situation easier.

Fuzzy logic might be useful for reasoning if the certainty factors on the premises are well chosen. However, fuzzy logic is arbitrary in the sense that there are no objective ways to assign those values. In many cases statistics or other probabilistic approaches are more dependable than fuzzy logic [Sow99].

6.1.2 Fuzzy Logic and Taxonomies

Angryk and Petry [AP03] introduce a way of combining fuzzy logic and ontologies. They construct fuzzy *is-a* and *part-of* hierarchies according to definitions of Wordnet [Fel98]. As an example they use Russia which belongs both to Europe and Asia. Russia can be marked to be part-of Asia with the certainty of 0.75 and part-of Europe with the certainty of 0.25. The fuzzy operations that will be used in reasoning about the ontology are not discussed.

Akrivas et al. [AWA+02] present an interesting method for context sensitive semantic query expansion. In the method, user's query words are expanded using fuzzy concept hierarchies. An inclusion relation defines the hierarchy. The inclusion relation is defined as the composition of subclass and part-of relations. Each word in a query is expanded by all the concepts that are included in it according to the fuzzy hierarchy.

In [AWA+02] inclusion relation is of the form $P(a, b) \in [0, 1]$. The meaning of the relation is the following. The concept a is completely a part of b, and high values of the $P(a, b)$ function mean that the meaning of a approaches the meaning of b. Thus, the fuzziness is limited only to one direction of the hierarchy, and there is not a way to express the fact that Lapland is a partial part of a number of countries.

Widyantoro and Yen [WY02] have created a domain-specific search engine called PASS. The system includes an interactive query refinement mechanism to help to find the most appropriate query terms. The system uses a fuzzy ontology of term associations as one of the sources of its knowledge to suggest alternative query terms. The ontology is organized according to narrower-term relations. The ontology is automatically built using information obtained from the system's document collections.

The fuzzy ontology of Widyantoro and Yen is based on a set of documents, and works on that document set. However, in this thesis the focus is on building taxonomies that can be used, in principle, with any data record set. The automatic creation of ontologies is an interesting issue by itself, but it is not considered in this thesis. At the moment, better and richer ontologies can be built by domain specialists than by automated methods.

One limitation that is related to all of the approaches above, when compared to the method presented in this thesis, is that the fuzziness works only in one direction of the concept hierarchy. In the work of Akrivas et al. [AWA$^+$02], the taxonomy is a crisp subsumption hierarchy in one direction and the fuzzy values only indicate how much of the meaning of the superconcept is covered by the subconcept. In the approach of Angryk [AP03], degrees of subsumption are represented, but there is no information about the portion of the superconcept that is covered by the subconcept. If one wants to represent fuzziness in both directions of the taxonomy, then fuzzy values have to be given in both directions. In the method presented in this thesis, overlap values are computed between any two concepts in a taxonomy, while the partial overlap values have to be given only in one direction. The coverage is determined based on the masses of the concepts.

In addition, the representation of disjointness between concepts of a taxonomy seems to be difficult with the tools of fuzzy logic. For example, the relationships between Lapland, Russia, Europe, and Asia are very easily handled probabilistically, but in a fuzzy logic based taxonomy, this situation seems complicated. There is not a readily available fuzzy logic operation that could determine that if Lapland partly overlaps Russia, and is disjoint from Asia, then the fuzzy inclusion value between Europe and $Lapland \cap Russia$ is 1 even though Russia is only a fuzzy part of Europe.

6.2 Nonmonotonic Logic

Classical logic is monotonic. Adding new axioms to a theory monotonically increases the number of theorems that can be proved. Nonmonotonic logic allows new information to increase or decrease the number of conclusions that can be derived.

Information that Polly is a penguin blocks the default conclusion that Polly can fly as a bird.

Unlike fuzzy logic, nonmonotonic logic depends on discrete changes in context rather than a continuous range of certainties. With nonmonotonic logic things can be grouped in a series of discrete levels of membership. Chairs, for example, can be grouped to typical chairs, somewhat untypical chairs, and things that are not chairs but resemble chairs in some way. However, the question, which of two untypical chairs is more typical, is meaningless without a context. In one context a wheel chair is more typical than rocking chair but in another context the result might be the opposite.

6.2.1 Nonmonotonic Logic and Taxonomies

The oldest, simplest, and most popular way of dealing with defaults is to tag properties of concepts with default values. The defined defaults can be overridden by subconcepts. For example, we define Bird as a concept and give it a property flies with the default value true. Then we define Penguin as the subconcept of bird and override the value of flies to false. This is a simple way of expressing vague knowledge about things.

In principle, the concepts of taxonomies could be extended with default values that could be overridden by subconcepts. However, the technique is good only when one can define the default values for a concept. It is easy to come up with examples where this approach does not work. For example in geographical ontologies based on part-whole relations it would be difficult to come up with the appropriate default values.

Another problem is that, the world is full of exceptions. How could we determine which values are more important than others? This could lead to difficulties in constructing the ontologies. Another problem is the algorithmic complexity of these nonmonotonic systems. Nonmonotonic systems are recently more and more overridden by probabilistic and fuzzy logic systems.

6.3 Rough Sets

Rough set theory is a mathematical approach to vague and uncertain data analysis [Paw82]. Rough set theory is based on the assumption that objects are perceived by means of the information about them, encompassed in a set of available features or attributes. This idea leads to the notion of information system which is a central concept in rough set theory. The information system can be seen as a data table, where each row represents an object, and each column represents an attribute. The objects are known only by their attributes.

More formally, an information system is a pair $I = (U, A)$ where U is a finite set called the universe of objects and A is a finite set of attributes. As a consequence of the basic assumption mentioned above, some objects may become indiscernible. The objects $x, y \in U$ are B-indiscernible if $B \subset A$ and x and y have the exactly same attribute values for all the attributes in B. In other words, x and y are B-equivalent, and $[x]_B$ is the B-equivalence class that x belongs to. A subset of the universe U of objects is called a concept.

For a concept $X \subseteq U$, two approximations can be defined relative to a set $B \subseteq A$:

$\underline{B}X = \{x \in U \text{ , D} = [x]_B,\ D \subseteq X\}$ and $\overline{B}X = \{x \in U : D = [x]_B\ ,\ D \cap X \neq \emptyset\}$.

$\underline{B}X$ is called the lower approximation of X and $\overline{B}X$ is called the upper approximation of X. $BN_B(X) = \overline{B}X - \underline{B}X$ is called the B-boundary region of X. If $BN_B(X)$ is empty, the X is said to be B-exact, otherwise X is B-rough. $\underline{B}X$ collects all those objects that belong certainly to X. The boundary region collects those objects which are vague with respect to X, i.e., have representatives both in X and in the complement of X.

In terms of the set theoretic approach of this thesis, if A is a concept, then the concepts that are subsumed by A belong to the lower approximation of A. Those concepts that partially overlap A belong to the upper approximation of A.

The rough sets approach provides an interesting view on the vagueness of concepts and relations between concepts. However, when considering ontologies of the semantic web, the problem is that rough set systems are based on the attributes of concepts. On the semantic web, taxonomies are often created without specifying any properties (attributes) to the concepts. Thus, a great deal of extra work will be needed if one wanted to create taxonomies capable of representing partial overlap based on rough set theory.

Another limitation is that the coverage between concepts still can not be expressed with rough set theory: If B belongs to the lower approximation of A, we still do not now, how much of A is covered by B.

Rough set theory would be a very interesting method if we would like to develop an automatic ontology creator based on given data. Then it could, in theory, create ontologies with a lot of encoded knowledge about partial inclusions. The goal of this thesis, however, is not to develop an automatic ontology creator, but only a way to represent partial inclusions in ontologies and form the *overlap tables* of concepts based on these profiles.

6.4 Rough Location

Rough location is a formalism developed by Thomas Bittner [Bit99] for characterizing the location of a spatial object within a set of regions which form a regional partition of space.

Spatial objects are objects that have a location in space. An important aspect characterizing what the spatial objects are is their compositional structure, which is the relationships between the whole object and the different parts comprising the object. This basic relation is formally defined as $P(x, y)$, which means that x is a part of y.

In rough location theory, location of an object is defined in terms of relations between spatial objects and regions of space. It is assumed that location can be characterized by relations between things that exist and one or more regions in which they are located. Location relates the compositional structure of spatial objects to the compositional structure of spatial regions. The spatial regions form a partition of the geographical space. According to Bittner, location can be defined in terms of exact location, part location, or rough location.

Part location is comprised of a number of locational relations. This means that there are multiple ways in which parts of objects relate to parts of regions of space. A part x can be fully located (FL) in y, or (OL) overlap located in y or it can exterior (NL) to y. For example, Lapland is overlap located in Norway, Sweden, Finland, and Russia.

The rough location of the spatial object o, within the set of regions forming the regional partition, G, is characterized by an n-tuple of relations. Those relations characterize the part location of the single spatial object, o, with respect to all elements, g, of the regional partition G.

Rough location can be used to deal with the indeterminacy of location caused by vagueness of object definitions. Consider for example the spatial objects mountain and valley. Where does the valley end and the mountain begin? The vagueness of the definitions of these concepts causes indeterminacy of their location. Indeterminacy of location means that there are multiple candidates of exact-location-regions which are consistent with the objects definition. A vaguely defined object, o, is located within a regional partition consisting of the three concentric regions 'core' (FL), 'wide boundary' (OL), and 'exterior' (NL). These regions resemble lower approximation, higher approximation and exterior of the rough set theory respectively.

According to Bittner [Bit99], the notion of rough location in regional partitions seems to play an important role in human cognition in general, so rough location could possibly be implemented in other uncertain domains than spatial location as well. From a cognitive point of view, rough location can be seen as a way of understanding an object as a whole by considering how its parts relate to a frame of reference with a known structure. The frame of reference is given by the underlying regional partition.

The idea of modeling concepts using a Venn diagram comes close to the ideas of Bittner. In this thesis, however, set theory was seen as a sufficient formalism to model the overlap between concepts. The main difference between the approach of Bittner and the method described in this thesis is that Bittner is not interested in constructing concept hierarchies, but just in defining the location of spatial object within a set of regions. In this thesis the focus is modeling uncertainty in concept hierarchies, and the geographical example is just a special case.

6.5 Probabilistic Reasoning

Zhongli Ding and Yun Peng [DP04] are creating a probabilistic extension to Ontology language OWL. Their idea is to increase the expressive power of ontologies by additional probabilistic information. In their approach, the OWL language is first augmented to allow additional probabilistic markup so that probability values can be attached to individual concepts and properties as well as their relations in an OWL ontology. Second, a set of translation rules is defined to convert this probabilistically annotated ontology into a Bayesian network.

According to the authors, the translation is plausible both because of the rigorous and efficient probabilistic reasoning capability and also because of the structural similarity between the DAG of a BN and the RDF graph of OWL. Their methods are based on probabilistic extensions to description logics [KLP97, GL02].

6.5.1 The Transformation Rules

The Bayesian network topology is generated according to a number of rules. All classes are translated into nodes of a BN, and an arc is drawn between two nodes in a BN only if the corresponding two classes are related by a predicate in the OWL file. The subClassOf property is translated into an arc from the superclass to the subclass.

Also the properties are mapped into nodes. A node represents the set of individuals in the domain who have this property. This node is a child of the domain class of the property. Every property node has a range node as its child. It also could have a restriction node. The properties hierarchies of OWL are generated also to the BN. A directed arc is constructed between two mutually exclusive or disjoint concept nodes.

If there is no arc between two independent concept nodes, they should also be d-separated with each other; there is also no arc between two implicitly dependent concept nodes, although they are not be d-separated with each other.

6.5.2 Comparison of Approaches

Besides the obvious similarities to the method described in this thesis, there are a number of clear differences between the approaches.

1. The aim of Ding and Peng is to create a method to transform any OWL ontology into a Bayesian network. The goal of this thesis is not to transform existing ontologies into Bayesian networks, but to create a method by which overlap between concepts could be represented and computed from a taxonomical structure. However, the graphical notation of this thesis and its RDF(S) implementation are designed so, that it is possible, quite easily, to convert an existing crisp taxonomy to an overlap graph.

2. In the approach of Ding and Peng, probabilistic information must be added to the ontology by the human modeler that needs to know probability theory. In the method described in this thesis, the taxonomies can be constructed without expert knowledge of probability theory or Bayesian networks.

3. The created Bayesian network in their approach is the goal of the work. In the method of this thesis, the Bayesian network is merely a background tool used to help in overlap modeling and query matching.

4. There is no explicit way to express the partial inclusions between concepts, but they have to be given through the conditional probability tables in the approach of Ding and Peng. In the graphical notation presented in this thesis, the partial inclusions are given explicitly by the user.

In principle, the transformation rules of Ding and Peng [DP04] could have been used for transforming the OG into a Bayesian network. In the following, this transformation is outlined. It is done to explain why the transformation rules of Ding and Peng were not adapted in the method described in this thesis.

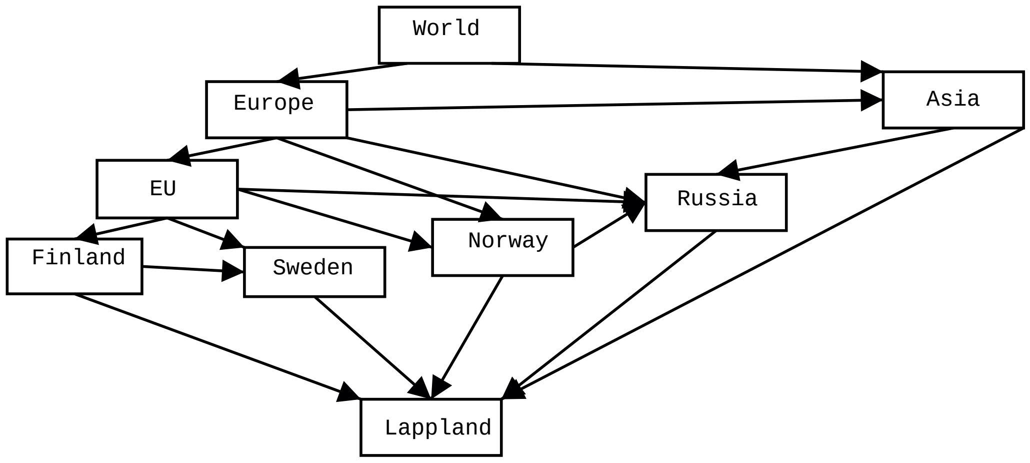

Figure 14: The topology of the alternative Bayesian network.

The probabilistic interpretation of the Venn diagram (i.e. the overlap graph) is given in the following:

1. The individual elements of the Venn diagram (e.g. pixels on the map), are interpreted as a population. Each individual in the population belongs to one or more groups. These groups are the sets of the Venn diagram, i.e. concepts of the taxonomy. Each group is represented by a boolean random variable in the Bayesian network.

2. The set of elementary outcomes is the set of samples of size 1 taken from the population. Thus the number of elementary outcomes equals the number of the individuals in the Venn diagram.

3. If A is a boolean random variable, then $P(A)$ is the probability that an individual taken as a random sample belongs to the group A. In effect $P(A) = \frac{|s(A)|}{|V|}$,

where $|V|$ is the number of elements in the Venn diagram, i.e. the number of atomic events.

4. A conditional probability $P(A|B)$ is the probability that an individual taken as a sample from the group B, belongs also to the group A. Thus, $P(A|B) = \frac{|s(B) \cap s(A)|}{|s(B)|}$, in terms of the Venn diagram the value indicates how much of B is covered by A.

Based on the above, all the variables (concepts) in the Venn diagram will get a prior probability resembling the relative size of the corresponding set. The variable representing the root concept of the overlap graph will get a prior probability one. According to the definition of conditional probability, if in a Bayesian network created from the the overlap graph of figure 6, Finland was selected, Sweden should get the probability 0. Because Sweden has a prior probability greater than zero, the disjointness between Finland and Sweden should be represented as an arrow in the Bayesian network. Because of this, Finland and Sweden can not be selected at the same time.

The topology of the Bayesian network can be created from the overlap graph by reversing the direction of each arrow in the diagram, and by adding an arrow indicating disjointness, between each pair of siblings in the network. The direction of the arrows indicating disjointness does not matter. For example, in the overlap graph of figure 6, arrows should be added between EU and Norway, Norway and Russia, and EU and Russia. The whole Bayesian network is presented in figure 14.

The CPTs for a variable A can be constructed in the following way. If A has no parents, then $P(A) = \frac{m(A)}{|V|}$. Thus, the root node gets a prior probability of 1. If A has parents, then go through all the value combinations (true and false) of the parent variables, operate according to algorithm 5 for each one. In the algorithm, C is the processed concept, VCP is the value combination of the parent variables, and CPTValue is the *true* value of the entry in the CPT.

The *false* values are computated be subtracting the *true* probability from 1. The overlap table of Lapland created by this BN is presented in table 4.

In this method, there is no need to create additional nodes to compute the overlap values. However, additional arcs have to be added.

Selected	**Referred**	**Overlap**
Lapland	World	1
	Europe	1
	Asia	0
	EU	0.6154
	Norway	0.3077
	Sweden	0.3077
	Finland	0.3077
	Russia	0.0769

Table 4: The *overlap table* of Lapland created by the alternative Bayesian network.

```
switch VCP do
    case more than one parent is true in VCP
        CPTValue = 0;
    case exactly one parent B is true in VCP
        CPTValue = (true: $\frac{|s(C)\cap s(B)|}{|s(B)|-|s(B)\cap(\bigcup\{s(A):A=false\wedge A\in VCP\})|}$);
    case no true parents in VCP
        CPTValue = 0;
end
```

Algorithm 5: The algorithm by which each entry in a CPT is constructed.

From the point-of-view of sorting hits according to overlap information about concepts, the posterior probabilities returned by this Bayesian network are not as good. If, for example, we wanted to create the overlap table, as defined in chapter 1, for Lapland, we would have to select each of the other concepts in the taxonomy one by one, and check the posterior probability for Lapland for each selection.

Another problem is that this Bayesian network does not enable the selection of two disjointed concepts at once. Thus, if a user is interested in Finland and Sweden, he will have to separately search for material about each of the countries. With the Bayesian network used in this thesis, they can be selected at once.

7 Discussion

A method to model uncertainty in semantic web taxonomies was presented. The method enables the modeling of the inexact, i.e. the non-crisp nature of the relationships between concepts. The method consists of a notation by which relationships between concepts can be represented, and a computing mechanism to reason based on this representation.

The notation and the overlap calculation method are based on crisp set theory. It is acknowledged that this is a simplification of the state of affairs in the real world. The use of fuzzy logic as the mathematical basis for the approach would have, at least in some cases, probably represented the outside world more realistically. In some situations it would be natural to let the partial subsumption values exceed 1. For example, an icon could be marked to be a utility article with the fuzzy value 0.7 and a work of art with the fuzzy value 0.8. In the method of this thesis the sum of these values have to be normalized to 1. The decision to use set theory and probabilistic calculus, however, has the following benefits when compared with the other formalisms:

1. The calculations are simple, but still enable the representation of overlap and vague subsumption between concepts.

2. By using set theory on probability calculus it is fairly easy to verify the correctness of the method.

3. The Bayesian representation of a taxonomy is argued in this thesis to be valuable, because it can be used also to other tasks than just the matching problem discussed in the introduction.

4. In some cases the set theoretic approach leads to better representation of the world than, for example, fuzzy logic (consider, for example, the geographical taxonomy presented as an example case in this thesis).

In this thesis, concepts are viewed mainly from the extensional point of view. A concept is seen as the set of entities belonging to it. Concepts can also be viewed from an intentional point of view. In this view, a concept is defined based on its features and not the entities belonging to it. There are situations in which a relation between concepts seems different, depending on whether they are viewed from the extensional or the intensional point of view. For example, icons share some features with utility articles, and some with works of art. Thus, from the intentional point of view, the concept *icon* is partially subsumed by both of these upper concepts. Based on the extension the concept there is no partial overlap, because each icon is both a utility article and a work of art.

However, it is still possible to represent this situation of intentional partial overlap in a Venn diagram. The overlap values could also be determined based on statistical data, by taking a random sample of N people, and asking each of them, whether an icon is a utility article or a work of art. The overlap value could be determined based on the frequencies of the different answers. Thus, the method presented is not limited, in principle, to the cases of extension based overlap.

Next the graph notation and the overlap calculation methods will be discussed separately.

7.1 Representing Overlap

In chapter 3 a graph notation was defined, by which partial subsumption and concepts can be represented in a quantified form. It was proved to enable the representation of any Venn diagram. However, there are some set structures that are very difficult to represent in practice, as the one in figure 7. However, in realistic concept modeling situations, these situations are rare.

In order for the notation to be useful, it should be easy to implement. The easiness of the RDF implementation can be considered from two different view point of views. First is the view point of conceptual simpleness, i.e. how simple it is to understand the notation and the meaning of different elements contained in it. As the number of elements of the language is small, and as the language is based on basic set theory, it is fair to say that the notation is conceptually simple.

Second is the view point of simplicity of use. The usability depends on the implementation of the notation. Thus, the usability can not be evaluated based on the structure of the notation only. However, the implementation of the notation, given in chapter 5, enabled the building of taxonomies with the graphical ontology development tool Protege[3].

[3]http://protege.stanford.edu/

7.2 The Computation of Overlap Values

The calculation of the overlap values between concepts is done by transforming the overlap graph into a Bayesian network. In principle, the calculations could have been done without the Bayesian network, however, it was chosen from the following reasons.

1. As the goal was to create a method for calculating the overlap values between a selected concept and every other concept in the taxonomy, the well known algorithms of evidence propagation over a Bayesian network offered effective tools.

2. The Bayesian network created in the process can be used also in other tasks than in the basic concept matching problem described in this thesis. For example, it could be used as a basis for personalization: If we had a user profile described in terms of the concepts of the taxonomy, we could enter it as evidence to the Bayesian reasoner, and it would suggest relevant material to the user. Bayesian networks have been used as a tool in user modeling [KSO$^+$01].

The Bayesian network that is created with the presented method is only one of the possible Bayesian networks that can be created based on the taxonomy. This one was chosen, because it answers to the problem statement of this thesis in the most direct manner, and because it is constructed easily based on the solid path structure. Moreover, in the used Bayesian network, disjointness between concepts is represented with d-separation. This is a valuable characteristic, as it enables the selection of two disjoint elements at once. For example, the user might be interested in Spain and France, and because of the structure of the network both could be selected at once.

The created Bayesian network does not calculate the posterior probabilities exactly by the definition of the conditional probability, but the inexactness is very small, and it should not cause any problems in practice.

7.3 Future Work

In the future we plan to test the presented method in practice in an ontology based search engine. We also intend to apply the above method to non-geographical taxonomies, for example to the domains of art and culture.

Also the refinement and further development of the graphical notation and its RDF(S) implementation will be considered to enhance its usability. The transformation of the taxonomy to another Bayesian network structures is yet another goal for future work, as well as the problem of using Bayesian networks as a basis for personalization.

References

AP03	R.A. Angryk and F.E. Petry. Consistent fuzzy concept hierarchies for attribute generalization. In *Proceeding of the IASTED International Conference on Information and Knowledge Sharing (IKS' 03)*, pages 158–163. ACTA Press, 2003.
AWA$^+$02	G. Akrivas, M. Wallace, G. Andreou, G. Stamou, and S. Kollias. Context-sensitive semantic query expansion. In *Proceedings of the IEEE International Conferrence on Artificial Intelligence Systems (ICAIS)*, pages 109–114. IEEE Computer Society, 2002.
BG	Dan Brickley and R.F. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema.* URL: http://www.w3.org/TR/rdf-schema/.
Bit99	Thomas Bittner. *Rough Location.* PhD thesis, Institute of Geoinformation, Technical University, Vienna, Austria, 1999.
BLHL01	Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web, a new form of web content that is meaningful to computer will unleash a revolution of new possibilities. *Scientific American*, 284(5), 2001.
Cha	Pierre-Antoine Champin. RDF tutorial. URL: http://www.710.univ-lyon1.fr/ champin/rdf-tutorial/rdf-tutorial.html.
DP04	Zhongli Ding and Yun Peng. A probabilistic extension to ontology language owl. In *Proceedings of the Hawai'i Internationa Conference on System Sciences.* IEEE Computer Society, 2004.
Fel98	Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* The MIT Press, 1998.
FF01	F. V. Finin and F. B. Finin. *Bayesian Networks and Decision Graphs.* Springer-Verlag, 2001.
GC03	V. Geroimenko and C. Chen, editors. *Visualising the Semantic Web: XML-based Internet and Information Visualization*, pages 36–48. Springer, London, 2003.
GL02	R. Giugno and T. Lukasiewicz. P-SHOQ(D): A probabilistic extension of SHOQ(D) for probabilistic ontologies in the semantic web. INFSYS Research Report 1843-02-06, Technische Universität Wien, 2002.

Gru93 T. R. Gruber. A translation approach to portable ontology specification. Technical Report KSL 92-71, Knowledge Systems Laboratory, Computer Science Department, Stanford University, 1993.

Haa96 Susan Haack. *Deviant Logic, Fuzzy Logic.* University of Chicago Press, Chicago, 1996.

Jen96 Finn V. Jensen. *An introduction to Bayesian Networks.* UCL Press, 1996.

KLP97 D. Koller, A. Levy, and A. Pfeffer. P-CLASSIC: A tractable probabilistic description logic. In *Proceedings of AAAI-97*, pages 390–397. AAAI Press, 1997.

KSO$^+$01 A. Kuenzer, C. Schlick, F. Ohmann, L. Schmidt, and H. Luczak. An empirical study of dynamic bayesian networks for user modeling. In R. Schafer, M.E. Muller, and S.A. Macskassy, editors, *Proc. of the UM'2001 Workshop on Machine Learning for User Modeling*, pages 1–10. Springer-Verlag, 2001.

LS Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification.* http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

Men00 Jerry M. Mendel. *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions.* Prentice Hall PTR, 2000.

NM01 Natalya F. Noy and Deborah L. McGuinnes. Ontology development 101: A guide to creating your first ontology. Technical Report KSL 01-5, Knowledge Systems Laboratory, Computer Science Department, Stanford University, 2001.

Paw82 J. Pawlak. Rough sets. *International Journal of Information and Computers*, 11:341–356, 1982.

PH01 Judy Pearsall and Patrick Hanks, editors. *The New Oxford Dictionary of English.* Oxford University Press, 2001.

RN03 Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*, pages 462–536. Prentice Hall, 2nd edition, 2003.

Sow99 John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks Cole Publishing Co., 1999.

SV00 H. Stuckenschmidt and U. Visser. Semantic translation based on approximate re-classification. In *Proceedings of the Workshop on Semantic Approximation, Granularity and Vagueness, Workshop of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 2000.

SWM Michael K. Smith, Chris Welty, and Deborah L. McGuinnes. *OWL Web Ontology Language Guide*. URL: http://www.w3.org/TR/2003/CR-owl-guide-20030818/.

WAS03 M. Wallace, G. Akrivas, and G. Stamou. Automatic thematic categorization of documents using a fuzzy taxonomy and fuzzy hierarchical clustering. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2003)*. IEEE Computer Society, 2003.

WY02 D.H. Widyantoro and J. Yen. A fuzzy ontology-based abstract seawrch engine and its user studies. In *The Proceedings of the 10th IEEE International Conference on Fuzzy Systems*. IEEE Computer Society, 2002.

Zad65 Lofti Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

Appendix 1. RDF(S) Implementation of the Graphical Notation

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY a 'http://protege.stanford.edu/system#'>
 <!ENTITY uncertain 'http://uncertain.com/uncertain#'>
 <!ENTITY kb 'http://protege.stanford.edu/kb#'>
 <!ENTITY rdfs 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
 xmlns:a="&a;"
 xmlns:uncertain="&uncertain;"
 xmlns:kb="&kb;"
 xmlns:rdfs="&rdfs;">
<rdfs:Class rdf:about="&uncertain;PartialSubsumption"
 rdfs:label="uncertain:PartialSubsumption">
<rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="uncertain;Mass"
rdfs:label="unceretain:Mass">
<rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="&uncertain;mass"
 rdfs:label="uncertain:mass">
<rdfs:domain rdf:resource="&rdfs;Resource"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&uncertain;overlap"
 rdfs:label="uncertain:overlap">
<rdfs:domain rdf:resource="&uncertain;PartialSubsumption"/>
<rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&uncertain;disjointFrom"
 rdfs:label="uncertain:disjointFrom">
```

```
<rdfs:domain rdf:resource="&rdfs;Resource"/>
<rdfs:range rdf:resource="&rdfs;Resource"/>
</rdf:Property>
<rdf:Property rdf:about="&uncertain;object"
 rdfs:label="uncertain:object">
<rdfs:domain rdf:resource="&uncertain;PartialSubsumption"/>
<rdfs:range rdf:resource="&rdfs;Resource"/>
</rdf:Property>
<rdf:Property rdf:about="&uncertain;subject"
 rdfs:label="uncertain:subject">
<rdfs:domain rdf:resource="&uncertain;PartialSubsumption"/>
<rdfs:range rdf:resource="&rdfs;Resource"/>
</rdf:Property>
</rdf:RDF>
```

Appendix 2. The Example Case RDF(S)

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY uncertain 'http://uncertain.com/uncertain#'>
 <!ENTITY kb 'http://protege.stanford.edu/kb#'>
 <!ENTITY rdfs 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'>
]>
<rdf:RDF xmlns:rdf="&rdf;"
 xmlns:uncertain="&uncertain;"
 xmlns:kb="&kb;"
 xmlns:rdfs="&rdfs;">
<rdfs:Class rdf:about="&kb;Place"
 rdfs:label="Place">
<rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="&kb;partOf"
 rdfs:label="partOf">
<rdfs:domain rdf:resource="&kb;Place"/>
<rdfs:range rdf:resource="&kb;Place"/>
</rdf:Property>
<kb:Place rdf:about="&kb;World"/>
<kb:Place rdf:about="&kb;Europe">
<kb:partOf rdf:resource="&kb;World"/>
</kb:Place>
<kb:Place rdf:about="&kb;Asia">
<kb:partOf rdf:resource="&kb;World"/>
</kb:Place>
<kb:Place rdf:about="&kb;Russia">
</kb:Place>
<uncertain:PartialSubsumption rdf:about="&kb;RussiaEurope"
 uncertain:overlap="0.1667">
        <uncertain:object rdf:resource="&kb;Russia"/>
<uncertain:object rdf:resource="&kb;Europe"/>
</uncertain:PartialSubsumption>
```

```
<uncertain:PartialSubsumption rdf:about="&kb;RussiaAsia"
 uncertain:overlap="0.8333">
<uncertain:subject rdf:resource="&kb;Russia"/>
<uncertain:object rdf:resource="&kb;Asia"/>
</uncertain:PartialSubsumption>
<kb:Place rdf:about="&kb;EU">
<kb:partOf rdf:resource="&kb;Europe"/>
</kb:Place>
<kb:Place rdf:about="&kb;Finland">
<kb:partOf rdf:resource="&kb;EU"/>
</kb:Place>
<kb:Place rdf:about="&kb;Sweden">
<kb:partOf rdf:resource="&kb;EU"/>
</kb:Place>
<kb:Place rdf:about="&kb;Norway">
<kb:partOf rdf:resource="&kb;Europe"/>
</kb:Place>
<kb:Place rdf:about="&kb;Lapland">
</kb:Place>
<uncertain:PartialSubsumption rdf:about="&kb;LaplandFinland"
 uncertain:overlap="0.3077">
        <uncertain:subject rdf:resource="&kb;Lapland"/>
<uncertain:object rdf:resource="&kb;Finland"/>
</uncertain:PartialSubsumption>
<uncertain:PartialSubsumption rdf:about="&kb;LaplandSweden"
 uncertain:overlap="0.3077">
<uncertain:object rdf:resource="&kb;Finland"/>
<uncertain:object rdf:resource="&kb;Sweden"/>
</uncertain:PartialSubsumption>
<uncertain:PartialSubsumption rdf:about="&kb;LaplandNorway"
 uncertain:overlap="0.3077">
<uncertain:subject rdf:resource="&kb;Lapland"/>
<uncertain:object rdf:resource="&kb;Norway"/>
</uncertain:PartialSubsumption>
<uncertain:PartialSubsumption rdf:about="&kb;LaplandRussia"
 uncertain:overlap="0.0769">
```

```
<uncertain:subject rdf:resource="&kb;Lapland"/>
<uncertain:object rdf:resource="&kb;Russia"/>
</uncertain:PartialSubsumption>
<uncertain:Mass rdf:about="&kb;MassWorld">
<uncertain:subject rdf:resource="&kb;World"/>
<uncertain:mass="851"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassAsia">
<uncertain:subject rdf:resource="&kb;Asia"/>
<uncertain:mass="414"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassEurope">
<uncertain:subject rdf:resource="&kb;Europe"/>
<uncertain:mass="345"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassEU">
<uncertain:subject rdf:resource="&kb;EU"/>
<uncertain:mass="168"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassRussia">
<uncertain:subject rdf:resource="&kb;Russia"/>
<uncertain:mass="342"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassNorway">
<uncertain:subject rdf:resource="&kb;Norway"/>
<uncertain:mass="36"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassFinland">
<uncertain:subject rdf:resource="&kb;Finland"/>
<uncertain:mass="36"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassSweden">
<uncertain:subject rdf:resource="&kb;Sweden"/>
<uncertain:mass="36"/>
</uncertain:Mass>
<uncertain:Mass rdf:about="&kb;MassLapland">
```

```
<uncertain:subject rdf:resource="&kb;Lapland"/>
<uncertain:mass="26"/>
</uncertain:Mass>
</rdf:RDF>
```