Diplomarbeit

# Modelling Ontologies with Topic Maps and OWL: Implementation Challenges and Conceptual Issues

ausgeführt am

Institut für Softwaretechnik und interaktive Systeme
Information & Software Engineering Group
der Technischen Universität Wien

unter der Anleitung von
ao.Univ.Prof. Dr. Andreas Rauber
und
Univ.Ass. Dr. Alexander Schatten
als verantwortlich mitwirkendem Assistenten

durch

Stefan Raffeiner
Matr.Nr. 9926599
Moissigasse 9/1/20, 1220 Wien

Wien, am 14. Juni 2005 _____

# Modelling Ontologies with Topic Maps and OWL: Implementation Challenges and Conceptual Issues

Ontologies provide a promising method of organizing and representing information and knowledge. Since the presentation of the Semantic Web vision in 2001, which is based on ontologies acting as "knowledge stores", significant amount of resarch work has been done in this field. This thesis deals with ontologies by focusing on the *Topic Maps* and *OWL* standards, which have become the most important formats for representing knowledge in ontologies.

Despite the relatively broad adoption of ontologies in the research community during the last years, creating and using ontologies is still no straightforward task. This is essentially due to the lack of proper tools that are easy enough to use in practice, but also due to some inherent limitations that apply to ontologies. Most issues related to ontologies pertain to one of these two groups of problems, which could be described as "implementation challenges" and "conceptual issues", respectively.

The thesis first introduces ontologies and related concepts like the Semantic Web and also presents an in-depth view of the Topic Maps and OWL standards. Three major topics are identified as being especially challenging when implementing ontologies in real world applications: providing persistent, scalable storage, performing complex queries on ontologies, and deducing implicit information by using special inference engines. Apart from these challenges on the implementation level, a number of crucial issues at the conceptual level of ontology representation formats are also discussed, such as the absence of contextual constraints, the inability to properly represent inconsistencies and the lack of methods for expressing uncertainty in knowledge bases.

# Modellierung von Ontologien mit Topic Maps und OWL: Implementierungsschwierigkeiten und konzeptionelle Fragen

Ontologien sin ein vielversprechender Ansatz, Informationen und Wissen zu organisieren und repräsentieren. Durch die Präsentation der Vision des Semantic Web im Jahre 2001, welches auf Ontologien als "Wissensspeicher" aufbaut, haben diese einigen Aufschwung innerhalb der wissenschaftlichen Gemeinde erfahren. Diese Diplomarbeit betrachtet Ontologien im Detail und konzentriert sich dabei auf die Standards *Topic Maps* und *OWL*, welche die beiden wichtigsten Repräsentationsformate für Ontologien darstellen.

Obwohl Ontologien in den letzten Jahren breite Anwendungsmöglichkeiten gefunden haben, ist das Erstellen und Verwenden von Ontologien nach wie vor nicht ohne Detailkenntnisse zu bewältigen. Gründe dafür sind einerseits das Fehlen geeigneter Software-Werkzeuge, welche in der Praxis einfach eingesetzt werden können, andererseits aber auch einige grundlegende Einschränkungen, denen Ontologien unterliegen. Die meisten Schwierigkeiten bei der Anwendung von Ontologien können demzufolge auf mangelnde Unterstützung beim Implementationsprozess als auch auf konzeptionelle Problematiken zurückgeführt werden.

Die vorliegende Diplomarbeit führt zunächst in die Grundlagen von Ontologien und verwandter Konzepte wie das Semantic Web ein und stellt die beiden Standards Topic Maps und OWL im Detail vor. Drei Themenbereiche werden dabei als besonders kritisch für reale Applikationen befunden: die persistente, skalierbare Speicherung von Ontologien, das Ausführen komplexer Abfragen über die Wissensbasis innerhalb einer Ontologie sowie das Ableiten impliziter Informationen aus den expliziten Fakten einer Wissenbasis unter Verwendung einer Inferenzmaschine. Neben diesen Implementationsschwierigkeiten wird weiters eine Reihe von Problemen aufgezeigt, welche mehr konzeptioneller Natur sind und dementsprechend kritisch zu bewerten sind. Dazu zählen vor allem das Nichtvorhandensein von kontextuellen Einschränkungen in Ontologien, die fehlenden Möglichkeiten zur Erfassung von semantischen Inkonsistenzen sowie das schwierige Abbilden von unsicherem Wissen.

# Danksagung

Die vorliegende Diplomarbeit wäre nicht ohne das Zutun vieler Menschen entstanden, denen ich an dieser Stelle meinen Dank aussprechen möchte. Meinen Eltern danke ich nicht nur für die finanziellen Zuwendungen während meines Studiums, sondern auch für ihre Motivierung und moralische Unterstützung. Großer Dank für Hilfestellungen während der Entstehung dieser Arbeit gebührt allen beteiligten Institutsmitarbeitern, insbesondere Alexander Schatten für seinen Einsatz, seine Zeit und die vielen interessanten Gespräche. Bei Prof. Andreas Rauber möchte ich mich für sein Engagement und die Bereitschaft zur Betreuung dieser Arbeit bedanken.

Nicht zuletzt möchte ich auch all jenen danken, mit denen ich in den letzten Jahren viel Zeit verbracht habe: meiner Schwester Martina, meinen Freunden und Studienkollegen Gerd, Markus, Matthias, Anton und Bernhard sowie meiner Freundin Sophie. Euch allen danke ich für die schönen gemeinsamen Stunden.

# Table of Contents

# List of Figures

# Introduction

In recent years, ontologies and related technologies enjoyed ever increasing popularity among researchers working in various areas usually related to knowledge management and knowledge representation. Not only theoretical foundations are being actively developed and investigated, but also a variety of practical applications has been implemented so far. Although ontologies are well known in the Artificial Intelligence community and thus can not be considered to be conceptually "new" inventions, the development of ontology-related standards (RDF, RDFS, OWL etc.) and adaption of existing ones (e.g. ISO Topic Maps) gained significant momentum in the context of the popular Semantic Web initiative. While the Semantic Web as originally envisioned is not likely to become reality in the near future (if at all), technologies focusing on ontologies actually exist and are used by many people in the scientific community. A large number of official standards and recommendations exists, as well as many tools ranging from simple ontology editors to advanced frameworks offering platforms for the development of ontology-centric applications.
Yet it is interesting to note that most of these standards and tools are not primarily concentrating on the realization of the Semantic Web itself, although they certainly benefit from the popularity of the term (or "buzzword", as many critics say). Instead, ontologies are commonly regarded as stand-alone technology, which, although also important for the Semantic Web, allows for many possible fields of application already on its own (i.e. without the additional components of the Semantic Web, see Section 1.2.3). For that reason, this thesis also regards ontologies as being conceptually independent from the Semantic Web, although existing relationships and possible influences are also investigated.
As mentioned above, the foundations and principles behind ontologies have been well known for a while. Roughly speaking, the key idea is to create an abstract model of a certain "domain of interest" by identifying categories and individuals within that domain as well as possible relationships. The proper connection of all these classes, instances, relationships and respective types is then said to be an ontology. This process however is not very different from what happens in the design phase of most software development projects and can for instance also be observed in the design of traditional relational databases, which usually represent classes with tables, properties with fields etc. The difference to the ontological approach is subtle at first sight: in an ontology, emphasis is clearly put on the *explicit representation* of arbitrary

concepts and relationships which has to occurr in a standardized way using a predefined *ontological vocabulary*. All implementation details such as performance considerations etc. are ignored on the level of ontologies, which are considered to be implementation-independent from a theoretical point of view.
One result from these considerations is that ontologies are not intended for *replacing* existing information containers such as databases, but rather form a separate, superimposed level of information representation that will extend the repertoire of tools for "knowledge workers": the considerable standardization efforts taken for ontologies, while assuring technical interoperability on the encoding and interpretation levels, also increase interoperability of complete "domain abstractions" on the level of human understanding. This means that creating an ontology can become a very appealing way for domain experts to exchange and share information about that domain in a concise, well-defined way. This is definitely not possible to such an extent for instance using a database and the corresponding EER diagram, because an ontology is inherently self-describing due to the standardized ontology vocabulary.
An quick and simple way to determine the maturity of any technology is to look at the real implementations that are used on a daily basis by actual users. For ontologies it turns out that so far they are quite successfully implemented whenever hierarchical and heavily interconnected classification schemas are already present "by nature". Examples include (public) library systems [Fit02, NZE05], hierarchical structures in product catalogs and product components [IM04] or systems related to software design [Bra04, dG03].
While such implementations can be considered to be prime examples for ontology-centric applications, the question arises to which extent ontologies can be used as universal, general-purpose containers for representing information and knowledge. A large part of this thesis will focus on this question by identifying implementation challenges that are encountered when developing software systems built around ontologies, presenting existing solutions and analyzing their applicability with real world systems in mind.

# 1 Ontologies and the Semantic Web

## 1.1 What is an Ontology?

The word "ontology" is being used in a number of different contexts and interpreted in many different ways. Hence some clarification about its meaning should be given first. While Section 1.1.1 gives some indications of what an ontology is, Section 1.1.2 confines frequently misinterpreted terms like "metadata", "taxonomy" or "thesaurus".

### 1.1.1 Definitions

The term "Ontology" can refer to two fundamentally different concepts, as for instance Webster's Third New International Dictionary states:

> "1. A science or study of being: specifically, a branch of metaphysics relating to the nature and relations of being; a particular system according to which problems of the nature of being are investigated; first philosophy.
>
> 2. a theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system."

While the first sense refers to a certain branch of philosophy, the second sense is used in Artificial Intelligence and Knowledge Representation and therefore also throughout this thesis. A more practical definition for "ontology" in this second sense is given by Tom Gruber:

> "An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of Existence. For AI systems, what 'exists' is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms

> that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory." [Gru95]

The term "conceptualization" is described by Gruber as "an abstract, simplified view of the world that we wish to represent for some purpose" [Gru95].
The definition given above contains two especially important observations: first, that an ontology usually covers only a certain domain of interest (or *universe of discourse*); second, that an ontology is *by definition* about *shared knowledge*. While both aspects seem to be perfectly acceptable, they turn out to be somewhat problematic with respect to real world applications, as it will be argued in Chapter 5.

### 1.1.2 Metadata, Taxonomies, Thesauri and Ontologies

In the last years, ontologies and ontology languages gained much attention from people all over the world, coming from very different branches of science and economy. This also entailed the imprecise usage of certain related terms, like "metadata", "taxonomy" or "thesaurus", which are often used incorrectly as synonyms for "ontology" or confused with each other. The most important differences are summarized in the following paragraphs; more detailed information can be found in Lars Marius Garshol's report "Metadata? Thesauri? Taxonomies? Topic Maps!" [Gar04a].

#### Metadata

The term *metadata* is being used for different purposes and signifies "data about data". Metadata can be any kind of data that gives additional information *about* another set of data. In the past, it was mainly used for describing schema information (e.g. structures of a relational databases) or administrative information (e.g. access rights etc.). A somewhat different approach is to enrich resources (documents, pictures etc.) with additional information like the name of the author of a document or the title of a picture, and label this "information about resources" metadata: "*going meta* does not mean necessarily going to an upper and more abstract structural level, it may mean going outside, after, before, besides or beyond, that is: in another semantic layer, to provide a description that is impossible to achieve from inside the reference layer" [AdMRV02].
A prominent example of metadata vocabulary[1] is Dublin Core [WKLW98] which allows for adding information like author, title, keywords etc. to a document. Dublin Core does *not* specify any syntax, nor does it define what the *contents* of any of the fields should be (e.g. there is no indication about how an author's name should be specified, as 'Stefan Raffeiner', 'S. Raffeiner' or 'Raffeiner, Stefan' etc.).

[1] A metadata vocabulary is the vocabulary the metadata itself *consists of* and must not be confused with the (controlled) vocabulary that may be used to control the *contents* of the fields defined by the metadata vocabulary.

### Controlled Vocabularies and Taxonomies

A *controlled vocabulary* is basically a predefined list of terms which can be used to classify resources, e.g. by using Dublin Core metadata. The purpose of such a list is to prevent authors from choosing terms that are too broad or too narrow for a specific domain or simply to avoid misspelling of terms.
A *taxonomy* is an extension of a controlled vocabulary that additionally arranges terms in a hierarchy. This makes it easier to find appropriate terms or to identify superordinate categories for terms. Except for the information expressed by the hierarchy itself, no additional information is added to the vocabulary (e.g. which terms are synonymous, deprecated etc.).
It should be mentioned that controlled vocabularies and taxonomies are not linked conceptually to metadata, but of course can be (and often are) used for populating metadata.

### Thesauri

*Thesauri* are in turn extensions of taxonomies and allow for expressing some relationships between terms by using a small vocabulary. There are two ISO standards regarding thesauri, ISO 2788 for monolingual and ISO 5964 for multilingual thesauri. According to these standards, thesauri may state that a term is a *broader*[2] *term* (i.e. it is one level higher in the hierarchy), or that a term is a *preferred term* for another term, or that a term is *related to* another term (but not strictly a synonym).
One of the best known thesauri for the English language is WordNet[3] which is described by its authors as "lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory". WordNet 2.0 contains a total of 203,145 word-sense-pairs, and allows to relate terms as synonyms[4], hypernyms[5], hyponyms[6], holonyms[7] and meronyms[8].

### Ontologies

The term *ontology* is often used to describe hierarchies that in reality are nothing more than taxonomies or thesauri; however, there is a substantial difference to these principles.

[2] This implies of course the existence of the inverse property "narrower term", which can be expressed directly by some thesauri.
[3] See [Wor] for the homepage of the project.
[4] A word which has the same meaning as another.
[5] A word that is more generic than a given word.
[6] A word that is more specific than a given word.
[7] A concept that has another concept as a part.
[8] A concept that is part of another concept. All definitions taken from [GLR].

In addition to the definitions given in Section 1.1.1 and with respect to the terms explained above, one can say that an ontology is "a model for describing the world that consists of a set of types, properties, and relationship types. [...] There is also generally an expectation that there be a close resemblance between the real world and the features of the model in a ontology" [Gar04a]. An ontology is therefore an extension to a thesaurus in a sense that thesauri, while having an open list of possible terms to classify, have a *fixed vocabulary* for expressing relationships between terms (i.e. "broader/narrower", "preferred" and "related to"); for ontologies, this limitation does not apply, as ontologies are capable of defining such relationships on their own. In fact, defining types of relationships between subjects is not different from defining types of subjects, which can be considered the main purpose of an ontology.
This observation shifts ontologies to a whole new level of expressiveness: an ontology is not only able to contain whatever terms seem appropriate, but is also able to store relationship types between terms *that can be defined by the ontology author* (see Section 2.5 for details). This is definitely not possible with thesauri or taxonomies. From a technical point of view, a thesaurus could be considered a simple ontology with three predefined relationship types, but in practice, thesauri are not referred to as ontologies due to their limited descriptive power. On the other hand, it is easy to represent thesauri with ontologies: e.g. Kal Ahmed designed a simple Topic Map ontology which is able to reproduce thesauri [Ahm03].

### 1.1.3 Representation Formats

While an ontology itself is a specification of a certain domain and therefore an *abstract theory*, a number of *representation formats* that can be used by computers to deal with ontologies has been defined, mainly in the area of Artificial Intelligence. In this context, the term "representation format" does not refer to a specific *serialization format* of an ontology language, but rather to technologies and standards used to represent ontologies; these standards may then have one or multiple notations for serialization purposes (see sections 3.1.2 and 3.2.1).
Most of these formats, however, never really left the AI corner and only a handful experienced wider adoption. Among these are also the following:

**KIF:** The Knowledge Interchange Format was created to support the transfer of knowledge between legacy knowledge-management languages. A simplified version of KIF (SUO-KIF) is used by the Standard Upper Ontology (see also Section 2.4.2).

**CGIF:** The Conceptual Graph Interchange Format is a normative serialization format of Conceptual Graphs that were introduced by John Sowa in 1976. "Conceptual graphs (CGs) are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence. Their purpose

> is to express meaning in a form that is logically precise, humanly readable, and computationally tractable." [Sow]

All statements that can be expressed by one of the above mentioned formats also can be expressed with the abstract syntax of the *Common Logic Framework* [Com] that has been submitted as a "new work item" to the ISO/IEC in 2001. The CL framework is a framework for logic-based languages and supports First-Order Logics (FOL).

Another important format for ontology representation is the proprietary language CycL which the Cyc Ontology (and its upper ontology subset OpenCyc, see Section 2.4.2) are formulated in. As its inventors claim, "CycL is a formal language whose syntax derives from first-order predicate calculus (the language of formal logic) and from Lisp. In order to express common sense knowledge, however, it goes far beyond first order logic" [Cyc02b].

With the rise of the World Wide Web, new representation formats built upon URIs and SGML/XML were considered. Currently, two different representation formats have experienced wider adoption: the International Standard ISO 13250 *Topic Maps* and the W3C Recommendation *Resource Description Framework (RDF)* with its ontological extension *Web Ontology Language (OWL)*. One consideration behind these new formats was, among others, to keep them rather simple (compared to fully-fledged FOL languages) and decidable[9] (and therefore computationally efficient).

The Topic Maps standard "has its roots in traditional finding aids such as back-of-book indexes, glossaries and thesauri" [Pep02b] and is usually not directly linked to ontologies, mainly due to its lack of an advanced ontological vocabulary. Also, Topic Map advocates claim that Topic Maps are "ontology-agnostic" [Vat03] and are therefore suited for representing any ontological vocabulary. The Topic Map standard is described in Section 3.1. Although there is no broad discussion about the relationship between Topic Maps and ontologies, some interesting efforts exist to establish Topic Maps as ontology containers. These considerations may actually be crowned with success due to some unique features that Topic Maps offer. Section 3.1.10 contains a in-depth discussion of the relationship between Topic Maps and ontologies.

RDF with OWL, on the other hand, "has its roots in formal logic and mathematical graph theory" [Pep02b] and has gained much popularity through the Semantic Web vision statement. A detailed description of the standard is given in Section 3.2.

[9] This is however not true without restrictions for the Topic Map standard, and also not for the fully-fledged version of OWL, OWL Full.

## 1.2 The Semantic Web

### 1.2.1 The Vision of the Semantic Web

When in 2001 Tim Berners-Lee published his vision of the Semantic Web [BLHL01], a lively discussion about the concept arose and is still ongoing. In this vision, Berners-Lee describes the principles of what he proposes as the successor of the World Wide Web as we know it today. The main point of the idea is to make the information on the WWW *understandable for computers* by creating "semantic annotations" to web pages in a computer-readable format, *in addition to the information that is presented to the human visitor of a web page*. Berners-Lee describes an advanced example of a personal agent that automatically schedules tasks for his owner based on information that it retrieves on its own from the Semantic Web, without requiring almost any human interaction. Berners-Lee gives even some indications about the technologies used to make such a future web possible, namely XML, RDF (see Section 3.2) and "the Semantic Web's unifying language", which didn't exist yet at the time of his writing, but can be assumed to be the Web Ontology Language standard OWL (see Section 3.2.3) released in 2004.
The main components of the Semantic Web, as presented by Berners-Lee, can be described as follows:

- Well-structured annotations on web pages (expressed with XML) extend the traditional WWW and enable agents to capture certain facts, such as information about people, their e-mail addresses or their phone numbers.

- These facts can be linked to further information contained in ontologies, which makes it possible to determine the "meaning" of the information and to deduce further information by using inference rules and inference engines.

- Concepts stored in ontologies can be identified by using URIs as identifiers; these URIs are used for electronic resources as well as for physical objects or abstract concepts. URIs also provide a way to create bridges from one ontology to another by linking certain concepts that exist in both ontologies.

- Agents [AJ01] are computer programs or scripts that act on behalf of a human (or organization) and are thus usually considered as electronic "personal assistants". The most significant properties of an agent, in contrast to other computer programs, are *autonomy* of action, *ability to communicate*, *adaptiveness*, *learning aptitude* and *mobility*[10]. Agents are prevalently used for retrieval and

[10] Mobility denotes and agent's ability to actively move between different computer systems that are connected by some kind of network; while mobility is a desirable feature, it is often hard to achieve, not only due to platform differences, and therefore regarded as non-critical for most applications.

processing of information, knowledge interchange, and executing tasks. Moreover, agents are suited for processing requests (from other agents) and often work on tasks in a distributed manner (e.g., by specializing on certain fields of activity). Within the Semantic Web scenario, agents are expected to carry out tasks by communicating and negotiating with each other, using ontologies and inference capabilities to establish a common vocabulary that is likely to change dynamically as needed.

- Finally, the need for trust and encryption is acknowledged, but not described in detail.

Since the publication of Berner-Lee's article, a remarkable number of people from different areas of research has been actively working on technologies that facilitate the creation of Semantic Web pages in many different ways. The World Wide Web Consortium (W3C) has released standards like RDF, RDFS, OWL etc. that the Semantic Web is built upon. A great number of usage scenarios and interesting ideas have been outlined, and a plethora of according software tools exist, not only for designing ontologies, but also for managing them and providing access to the information contained inside them. However, apart from a few so called "Semantic Web frameworks" such as Jena [Jen] and Sesame [BKvH01] which managed to attract a significant number of users, most tools must either be considered to be in an experimental stadium, or were intentionally designed as proof-of-concept only; many focus on technically experienced users such as software developers or Semantic Web researchers. Real world applications of Semantic Web technologies are scarce, if not inexistent, which leads to the conclusion that the Semantic Web initiative has not gained much momentum among "ordinary" users not directly involved in the research process.
Many critics of the Semantic Web refer to the original vision, which is indeed quite all-embracing and assumes a large number of components that are not realized yet, like agents, easy-to-use ontologies with inference engines and well-established trust-providing systems. But the main purpose of a vision is to draw a scenario that is hard to reach and yet not unreal, rather than to provide actual solutions, and the Berners-Lee's visionary description of the Semantic Web certainly lives up to this end.

### 1.2.2 Ontologies and the Semantic Web

With regard to the main components of the Semantic Web (as proposed in Berners-Lee's vision), ontologies and their representation formats are certainly the most interesting and maybe most critical ones: while creating structured annotations with RDF or using URIs for subject identification purposes seems a straightforward task (given some familiarity with the corresponding standards), creating ontologies implies a number of questions and problems that are not so easy to answer. It is important to point out that ontologies definitely form the back-end of the Semantic Web

and are a key component for its success. As stated earlier, the biggest achievement of the Semantic Web is the transition of content from *computer-presentable* data to *computer-understandable* information (i.e. from HTML etc. to ontologies). Only by making the *content* of web pages available for automated processing by machines, a certain degree of autonomous action can be achieved.
It is of course a dangerous statement to claim that computers are capable of "understanding" information in a way we humans do. Without going into the (philosophical) discussion on human perception of information, computer scientists often simply judge a machine's capability to understand information by its actions taken with respect to the goals targeted; or, as Gruber states, "an agent 'knows' something if it acts as if it had the information and is acting rationally to achieve its goals" [Gru95].
Computer agents must therefore be capable of recognizing and identifying concepts that are presented to them within web page annotations or ontologies, categorizing new concepts and relating them to existing, known ones, as well as deducing further knowledge from that categorization as needed to accomplish the respective tasks. These requirements are essential for any automated, autonomous processing of information and go far beyond the simple exchange of data via a certain network protocol (which is what the WWW is basically about). In this context, ontologies can be seen as the heart of the Semantic Web by enabling agents to meet those requirements.

### 1.2.3 Ontologies in the Layered Architecture

The important role of ontologies for the success of the Semantic Web is also visible in the famous "layered architecture" diagram, as shown in Figure 1.1, which was presented by Berners-Lee at XML 2000. Although only one layer is titled "Ontology vocabulary", ontologies and their representation formats are in fact present in all three layers "RDF + rdfschema", "Ontology vocabulary" and "Logic". Since at the time of writing there were almost no concrete solutions or even suggestions[11] available for for the two upper layers, "Proof" and "Trust", ontology-related technologies form the biggest part of the layered architecture.
In order to make the Semantic Web possible, there must exist feasible solutions for all layers in the diagram:

**Unicode, URI:** Unicode [UC,Kuh99] and URIs provide a system-independent way for encoding and addressing information. They are already used by the World Wide Web and also form the basis for the Semantic Web, together with network and transmission protocols like TCP/IP and HTTP.

**XML:** XML and its related standards provide a way to represent arbitrary information structures and are the de facto standard for data exchange between applications

[11] Apart from Berners-Lee's vague comment at XML 2000 to use Public Key Infrastructures for such purposes.

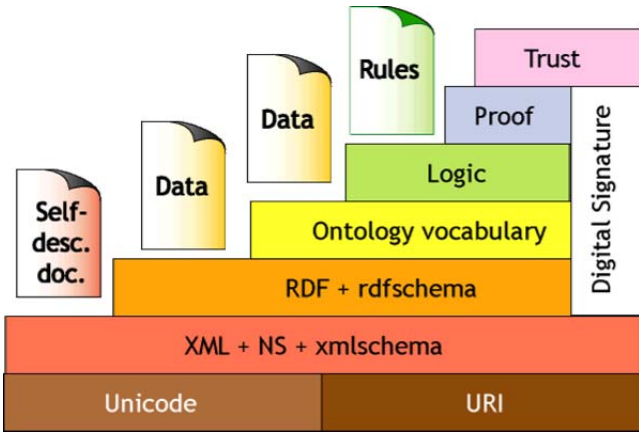


Figure 1.1: The Semantic Web Layered Architecture [BL00].

[Beh03]. Given their broad support of both tools and users, they are a reasonable choice for serving as a serialization format for ontologies.

**RDF:** RDF and RDFS provide a simple way to represent assertions as *information triples* that together build a graph of concepts and relations between them (see Section 3.2 for details). Thus, RDF and RDFS can be used as a format for representing the information contained in an ontology. The ISO-standard 13250 *Topic Maps* aims at the same goal, but uses a somewhat different approach and structures with increased complexity (see Section 3.1 for details).

**Ontology:** Neither RDF(S) nor Topic Maps provide a way to express certain advanced properties of concepts or relations (like cardinality constraints, transitivity etc.) that are needed for semantically qualifying the contents of the ontology, and that provide the basis for inference and logical rules. Many of these advanced properties are currently provided by the Web Ontology Language (OWL) which is built upon RDFS. For Topic Maps, support for inference is partly provided by several query languages, while formulating constraints may be facilitated by the upcoming standard TMCL (Topic Map Constraint Language).

**Logic:** On top of ontologies, inference engines are used to retrieve assertions that are not directly expressed within the ontology, but can be derived of existing facts. While usually not part of an ontology, inference engines need the input from an ontology to retrieve implicit facts needed to make computer agents behave"intelligently".

Two more key factors are not addressed by the layered architecture diagram: there must be a consensus on how to *establish identity* for arbitrary concepts (physical and abstract ones as well as electronic resources), and there has to be some kind of immediate benefit for potential ontology authors to create ontologies and share them with others via the Semantic Web. The latter issue will be discussed in Section 1.3, while establishing identity is a very complex problem that will be discussed in Chapter 2.

## 1.3 Ontologies and Knowledge Management

Knowledge and Knowlege Management are one of the most used buzzwords not only in the information industry, but also in very different research disciplines; they are also used for many different purposes and with varying connotations. Therefore it is almost impossible to provide a specific definition of Knowledge Management, as people with different backgrounds and interests obviously emphasize different aspects of "knowledge". For many organizations, managing, conserving and building up knowledge has become an important or even crucial part of their business activities, and very

often strong support from computers is needed, but often limited to some elementary document-organizing principles.
Regarding the terms "data", "information" and "knowledge", they are often used inconsistently and synonymously for each other: but "in order to understand the crucial issues of 'Knowledge Management', however, it is important to distinguish between them" [SEW01]. An informal definition of data and information is given by Küng et al.:

> "Data consists of facts and measurements that are represented through symbols for the purpose of processing and storage (usually highly structured, e.g., fields in relational database tables) without specification of the possible use. At this level, data is relatively meaningless to the user. When data is placed within a meaningful context, it becomes information. Information can therefore be regarded as data put in a specific problem context." [KLS+01]

"Knowledge" is generally interpreted as combination of information "with experience, context, interpretation and reflections and can therefore be regarded as a high-value form of information that is ready to apply to decisions and actions and may be of several types: Knowledge about something is referred to as declarative knowledge, procedural knowledge is knowledge of how something is performed, and knowledge dealing with why something occurs is called causal (or analytical) knowledge" [SEW01].
As explained earlier in Section 1.1.2, ontologies are a way to represent and store knowledge about a certain domain in a declarative manner (i.e., facts are directly represented by symbols and not encoded as procedural programs [Tau02]). This is the reason for considering ontologies (and related technologies) as central parts of advanced knowledge management efforts, as they are made by many organizations. The key advantage of an ontology is that the facts it contains can both be viewed and used by humans through an appropriate interface (which is of course also true for any electronic document, for instance) *and* at the same time utilized by computers (or maybe agents) without any human interaction at all. By opening organizational knowledge to existing and future computer agents, the initial investments will surely pay back at some point in time. But since ontologies can equally well be used directly by humans, an immediate benefit is additionally achieved.
In order to denote ontologies as enabling technology for computer-supported knowledge management, however, some aspects must not be left out. In addition to representing hierarchies of concepts and relationships between them, a number of additional requirements must be met by ontologies and their encapsulating ontology management systems: presenting an easily navigable structure, providing links to actual resources and versioning of information are only a few of them. Additionally, ontologies also provide a good solution for dealing with changing knowledge ("knowledge maintenance") because the traditional distinction between the schema layer and the actual content

(e.g., of a relational database) is almost inexistent (see Section 2.5). For ontologies, changing the schema (the class hierarchy) or parts thereof is as easy as changing the actual data.
Finally, ontologies are also a promising way to gain benefit from inference engines: by using an inference engine, implicit facts contained in the ontology can be retrieved. Without that implicit knowledge, an ontology can still be regarded as a useful, universal datastructure for storing facts, but it will be almost useless for agents that need actual "knowledge" as defined earlier. This is because some basic "intelligence" is expected from an agent which can only be obtained by also providing implicit facts in addition to the explicit facts that are present in the ontology (see also Section 4.3). A detailed description of the two prevailing standards for ontology representation, RDF/OWL and Topic Maps, and their respective capabilities is presented in Chapter 3.

Summarized, it can be said that ontologies are well suited for knowledge management purposes because of the following facts:

- They are built around *concepts* and represent "things" and relationships between them; this is basically what "knowledge" is built upon.
- They can be used by both humans and computer agents, which allows for great future evolution while providing immediate benefit.
- They can be used to provide input for important knowledge management features such as navigation, links to external resources etc.
- They support evolution and change of knowledge through their flexible organization of concepts in a variable "network" rather than using fixed schemas.
- They may be used by inference engines to retrieve implicit knowledge that is vital especially for computer agents.

This leads to the assumption that ontologies will probably be recognized as effective tools for supporting knowledge management within organizations even if there is little presence of the Semantic Web by now: ontologies are definitly promising tools on their own, even without the additional benefit of the Semantic Web. Indeed, since the number of ontology representation standards is very limited, interoperability between existing ontologies will not be hard to achieve. This can result in a strong tendency towards connecting ontologies, which is basically what the Semantic Web is about. Of course, some serious issues like scalability and security have to be taken into account if distributed ontologies are expected to grow as fast as the World Wide Web.

# 2 Concepts and Identity

The most important aspect of ontologies is certainly their ability to store facts about arbitrary concepts, which makes them promising tools for knowledge management. With respect to this functionality, a few important questions arise:

1. What is exactly a "concept"? How do terms like "subject" and "resource" relate to them?
2. How are concepts represented within an ontology? How are concepts linked to their corresponding counterparts in an ontology, i.e. how is identity established for concepts in an ontology?
3. How can different ontologies refer to the same concept and yet assure some level of interoperability?
4. How are concepts organized in an ontology?

Answers to these questions will be given in the following sections.

## 2.1 Concepts, Subjects, Resources

Ontology representations are so called "identity-based technologies" [Gar03a] because ontologies focus on "things", also called "concepts" or "subjects of discourse". Topic Maps always refer to concepts as "subjects"; a formal definition is given by the ISO 13250 standard:

> "In the most generic sense, a 'subject' is any thing whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever." [BBN02]

Or, as Steve Pepper states: "in short, a subject can be any *subject of discourse* that an author wishes to identify, name, represent, or otherwise make assertions about" [Pep03].
Likewise, RDF uses the term "resource" defined in RFC 2396 to refer to concepts:

> "A resource can be anything that has identity. [...] Not all resources are network "retrievable"; e.g., human beings, corporations, and bound books in a library can also be considered resources." [BLFIM98]

This definition is somewhat problematic, as the concept of identity is not defined anywhere in the standard. It is important to note that RDF "resources" are definitely not limited to electronic resources, but may also refer to any other existing or non-existing concept: "However, by generalizing the concept of a 'Web resource', RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web" [RPR04].

The concept of identity is quite fuzzy and the discussion about it has a long philosophical tradition. While identity is generally assumed to be the *sum of all characteristics of a certain subject*, this approach is difficult to handle for information technologies, because it has to be recognized that those characteristics may vary over time, without the identity getting lost. Some people even claim that there is no concept of "identity" at all, because even if some consensus about a certain subject of discourse is achieved, the communicating parties may still refer to slightly different concepts or have different views on it.

Establishing at least *some kind of* identity is the first step if two ore more parties wish to communicate about a subject: every party must know what the others are talking about, otherwise the communication makes no sense. Taking conversation among humans as the probably most prominent example, identity is usually established through "context, experience, and secondary information that help to round out the concepts that are being shared" [Deg04]. If the context is unclear to a participant of the conversation, further explanations or descriptions are needed until it is obvious that everybody is talking about the same "thing", or subject.

For machine communication, establishing identity is equally important. In contrast to humans, computers always need some kind of *unique identifier* or address, for a certain subject. Only by comparing such identifiers, the identity of a subject can be guaranteed. Examples for identifiers are paths and filenames of files on a hard disk, unique keys in database applications, or URLs of web pages[1]. For such resources, establishing identity is therefore straightforward[2] and can be achieved by comparing their respective identifiers: if any two resources have the same identifier, they are assumed to be identical. Establishing identity for subjects that are not electronic resources is more complicated, as explained in the next section.

[1] It is important to differentiate between the *concept of a certain web page as a whole* and the *contents of a web page*: e.g., someone might add the concept "Homepage of Apple Computer, Inc." to his ontology, and then assert things about this concept (e.g., that he last visited the site on a certain date). Opposed to that, the actual contents of the page, such as text, images etc., are not inherently tied to that concept – they appear if the page is viewed in a browser, but apart from that, there is no conceptual link between the homepage itself and its contents. URLs as unique identifiers (such as `http://www.apple.com`) can only be used for concepts, since contents must be considered dynamic and are likely to change.

[2] Actually, there is a number of problems with using URLs as identifiers for concepts; see Section 2.3.3 for details.

## 2.2 Representation and Identification of Arbitrary Subjects

### 2.2.1 Proxies as Binding Points for Subjects

The key purpose of an ontology is to make it possible to store facts about subjects. This is achieved by providing "*binding points* from which everything that is known about a given subject can be reached" [Pep03]. Binding points are therefore electronic placeholders, or *proxies*, for subjects. This is not only true for subjects that do not have an electronic locator by nature, such as objects of the real world or abstract concepts, but also for electronic resources that already have one or more locators. The advantage of this non-distinction between real world subjects and electronic resources is that they can be treated exactly the same way by an application or potential user. For Topic Maps, the proxies for subjects are called "topics"; RDF uses the term "node" (as in graph theory) to refer to a subject (see sections 3.1.3 and 3.2.1, respectively). Since the proxies for subjects in an ontology are electronic resources themselves, they are easily identifyable through some sort of locator, such as a numeric id, or URI (as defined in RFC 2396 [BLFIM98]) etc. Any assertion that is to be expressed by the ontology can use that identifier to refer to the proxy, and hence to the represented subject. This however says nothing about how the relationship between a subject and its proxies[3] is established; RDF and Topic Maps take somewhat different approaches here.

### 2.2.2 Identification with RDF

As for RDF, each "node" uses exactly one address (URI) to refer to a subject. If the subject is an electronic resource that can be retrieved through an URI (such as a webpage), that URI is used. If the subject is not an electronic resource, an arbitrary URI can be used to refer to that subject, although it is highly recommended to use an existing *public subject identifier* (see Section 2.3) if there exists one. In either case, two subjects are considered identical if their URIs are the same (i.e. if two URIs are identical, they refer to the same resource).

### 2.2.3 Identification with Topic Maps

As for Topic Maps, the relationship between a subject and its proxy can be created in two different ways, depending on the nature of the subject to be represented. As long as a concept in an ontology refers to an electronic resource, the identification method is the same as with RDF: the (XML-based) Topic Map notation XTM uses the URI of that resource, which is called *subject address* in this case, since the subject

[3] Any subject may have multiple different proxies, obviously, since even if only one proxy per ontology is allowed for a subject, there usually exist multiple ontologies that refer to the same subject of the real world.

is addressed directly through its locator. By comparing any two URIs, it can again be determined whether they refer to the same subject. The Topic Map Data Model proposal defines the term "information resource" for Topic Maps as follows:

> "An information resource is a resource that can be represented as a sequence of bytes, and thus could potentially be retrieved over a network." [GM05]

For subjects that can not directly be addressed by a computer, a *subject indicator* is needed to establish identity:

> "A *subject indicator* is a resource which provides some kind of compelling and unambiguous *indication* of identity of a subject to humans. [...] A subject indicator is different from the subject that it indicates." [Pep03]

The subject indicator itself, being an electronic resource, is computer addressable by definition; its address is comparable to the subject address (i.e. they are both URIs), but named *subject identifier*. The difference between a subject address and a subject identifier is that the subject address directly references the subject to be represented by the topic, whereas the subject indicator references a resource (the subject indicator) which gives an indication about the identity of the subject. The subject indicator is usually a resource whose content gives a human interpretable indication about, or description of, the subject that is to be represented. Examples of subject addresses and subject indicators can be found in Section 3.1.3.

## 2.3 Published Subjects

The concept of Published Subjects was introduced by the Topic Map community for interoperability between Topic Maps, but it is equally well suited to establish interoperability between Topic Maps and RDF/OWL. The specification for Published Subjects was created by the OASIS *Topic Maps Published Subjects Technical Committee* and gives the following indication about the purpose of Published Subjects:

> "Published Subjects [...] provide an open, scaleable, URI-based method of identifying subjects of discourse. They cater for the needs of both humans and applications, and they provide mechanisms for ensuring confidence and trust on the part of users. Published Subjects are therefore expected to be of particular interest to publishers and users of ontologies, taxonomies, classifications, thesauri, registries, catalogues, and directories, and for applications (including agents) that capture, collate or aggregate information and knowledge." [Pep03]

Esstblishing Published Subjects is considered to be an open and distributed process, as described in the next sections.

### 2.3.1 Creating Published Subjects

A subject indicator, as described in Section 3.1.3, is basically an electronic resource; every resource that is appropriate to establish the subject's identity if conceived by humans may be used as subject indicator. However, since the author of a resource does not necessarily know that the resource is used as a subject indicator, it must usually be taken into account that resources are either not stable (e.g. disappearing web pages) or unambiguous enough to be used as long-lasting subject indicators for subjects. To overcome this problem, a *publisher* deliberately publishes subject indicators for any subjects he may feel responsible or qualified. By doing so, these subjects become *Published Subjects*, their subject indicators become *Published Subject Indicators (PSIs)* and their addresses, the subject identifiers, become *Published Subject Identifiers (PSIDs)*. The publisher assures the stability and unambiguousness of these Published Subject Indicators and Identifiers.
In order to enable the publishing of subjects on a large scale and in a distributed environment, there are no criteria a publisher has to meet: anybody can act as a publisher for whatever subjects he wants to. It is up to the author of an ontology to decide which publishers and published subjects he trusts. Typically, a provider will not publish only one subject, but a whole collection of subjects all belonging to a certain domain; an ontology author may look for adequate published subjects provided by some reliable publisher, and if no appropriate published subjects exist, he may publish the subjects himself. Whenever a better provider for a subject can be found, it is easy to create a mapping from the self-published subjects indicators to others.
It should also be mentioned that "published" does not necessarily mean "public": published subject can also exist within an intranet or a single computer with no public access.
Several published subjects are already defined e.g. within the Topic Map standard itself (the subjects "topic", "association" etc.) or have been defined by Technical Committees at OASIS.

### 2.3.2 OASIS Vertical Domain Applications

In August 2001, the Organization for the Advancement of Structured Information Standards (OASIS) decided to form three new Technical Committees (TCs) in order to focus on vertical domain applications of the Topic Map paradigm:

**Topic Maps Published Subjects TC** to "promote the use of Published Subjects by specifying recommendations, requirements and best practices, for their definition, management and application."

**Topic Maps Published Subjects for Geography and Languages TC** to "advance

the XML Topic Maps specification for navigating information resources by defining published subjects for languages, countries, and regions."

**Topic Maps Vocabulary for XML Standards and Technologies TC** to "define a Topic Maps Published Subjects vocabulary for the domain of XML standards and technologies, which will provide a reference set of topics, topic types, and association types that will enable common access layers and thus improved findability for all types of information relating to XML, related standards, and the XML community."[4]

### 2.3.3 Subject Indicators for RDF

The main difference between Topic Maps and RDF with respect to establishing identity is how they relate to a subject that is not a network-retrievable resource: Topic Maps carefully distinguish whether a referenced resource *is the subject itself* (e.g. a topic representing a web page by means of a subject address), or whether the referenced resource is only a *indicator for a subject with no inherent address* (e.g. a topic representing a person by means of a subject indicator).
In RDF, there is simply no such distinction: "In RDF, the distinction between the RDF model and the world it represents is not given much emphasis, whereas in topic maps this distinction permeates the whole model" [Gar03a]. The absence of this distinction is actually a very fundamental limitation; in RDF, it is not clear if a RDF resource with URI

`http://www.raffeiner.at/about_me.html`

should represent a natural person or the web page that URI resolves to. Steve Pepper and Sylvia Schwab outline the problem in their paper "Curing the Web's Identity Crisis":

> "This indiscrimate use of URIs to identify subjects both directly and indirectly can be traced back to a lack of clarity regarding the very notion of "resource" in the Web community.
>
> Historically, in the model originally envisioned by Tim Berners-Lee, resources were simply documents (information resources) that had locators. Those locators turned out to be very useful as identifiers for documents - for example, when attaching metadata to them. But as the Web matured a need arose to be able to make assertions about things that were not documents (e.g., people and organizations) and the same practice of using URLs as identifiers was simply extended without much thought for

[4] All descriptions taken from `http://www.oasis-open.org`.

> the consequences. This in turn led to the more general notion of 'resource' as defined in RFC 2396 [...].
>
> The term 'resource' obscures the fact that some of the things we want to identify have locations and others do not. Using a 'locator' for something that has a location makes sense; using it for something that does not is asking for trouble." [PS03]

If there is no clarity about what an URI resolves to, the whole concept of identity-based technologies is almost useless because

- ontologies will not be reusable as they will be unable to unambigously identify the subjects of discourse;
- they won't certainly be able to interoperate, since there is no way of telling that two proxies refer to the same subject;
- therefore, it will be impossible for humans and agents to aggregate information and knowledge;
- this has the effect that ontologies will never scale beyond a closely controlled environment, which is directly opposed to the vision of the Semantic Web.

The solution for the problem is to extend the RDF/OWL standards to recognize the distinction between addressable and non-addressable subjects. Some people have supposed to use a special URI syntax (e.g. by using the # sign) to identify one of the two, which would eliminate the need for changing the actual RDF standard. Others argue that this "quick and dirty solution" only shifts the problem from resources to resource fragments (which the # sign is intended for). Whatever proposal is adopted, it will certainly be necessary to advance the development and usage of the Semantic Web; in fact, although the discussion may seem rather academic, one of the first questions of an ontology author is certainly which URIs he or she should use for identifying "things".

## 2.4 Classes and Instances in Ontologies

### 2.4.1 Subclassing and Instancing

Any subject of discourse can be considered to be an instance of a more general concept, or *class*, as shown in Figure 2.1. For example, Vienna is a concrete instance of the abstract concept "city" (as it is used in the statement "a city is a large and densely populated urban area" – this is not a statement about any existing instance of a city, but a statement about cities in general, i.e. the class of cities).

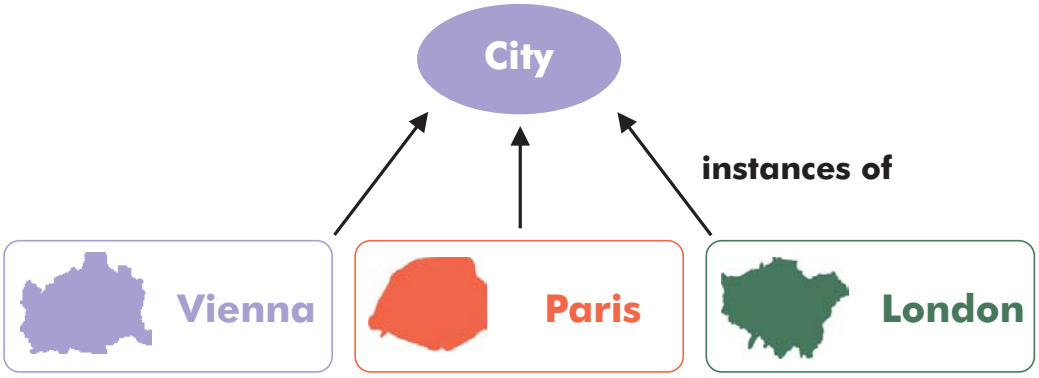


Figure 2.1: The class "city" and some instances.

Classes, on the other hand, can be *instances* or *subclasses* of other classes. For instance, the class "city" could be treated as a subclass of "settlement" and *at the same time* as an instance of "geographic concept". There is a substantial difference between the relationships "instance of" and "subclass of": regarding the example given above, it is correct to say "Vienna is a city" and "Vienna is a settlement"; it is however wrong to say "Vienna is a geographic concept"[5]. The subclass relationship is transitive, which means that if A is a subclass of B and B is a subclass of C, then also A is subclass of C. This brings with it that all instances of a class are also instances of all superclasses of that class. Figure 2.2 illustrates these relationships. While having subjects that are both classes and instances is perfectly correct from a theoretical point of view, it adds significant complexity to ontologies with respect to their computational complexity. This somewhat academic approach to classes can be summarized as follows:

1. Any subject can be the instance of one ore more general concepts, or *classes*.
2. Classes can be instances or subclasses of other classes, which results in a tree of classes, the *class hierarchy*.
3. Transitivity of the subclass relationship propagates all properties of a superclass to its subclasses.

Class hierarchies within ontologies are created with the purpose of classifying and organizing instances of subjects. However, ontologies are capable not only of creating trees and heriarchies of classes, but also of interconnceting the various classes and instances through other, arbitrary relationships, like "is part of" etc. The more relationships exist between the concepts in an ontology, the greater the potential of the ontology with respect to its expressiveness can be assumed.

[5] A similar example can be found in Rath's Topic Map Handbook [Rat03].

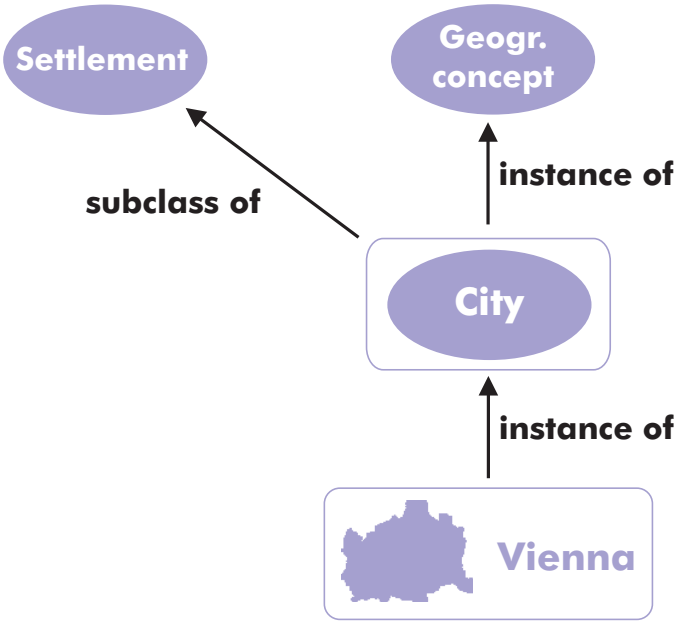


Figure 2.2: The concept "Vienna" as class and instance.

Sometimes, ontologies are regarded as *collections of classes* only, i.e. not containing any actual instances. As such, they are used as some kind of indices or organizing principles for "external" datasources, like traditional databases, or semi-structured resources like electronic documents etc. However, the factual non-existence of a clear distinction between "classes" and "instances" must be considered problematic for such ontologies and prohibits a universally valid definition of ontologies as "containers for classes". Throughout this thesis, ontologies are therefore assumed to *contain both classes and instances* (whenever that distinction applies).

### 2.4.2 Upper Ontologies

Since designing and populating ontologies is a very labor-intensive and time-consuming task, ontologies usually cover only a certain domain of interest in a more detailed way. It is however difficult to create an ontology from the scratch, since many concepts and rules from our everyday life are of course not present by default in an empty ontology. This makes it very difficult for agents to get reasonable results from an ontology, especially if an inference engine is to be used, because an empty ontology is like a "new universe" where no rules exist; by default, an inference engine can not distinguish between e.g. animate and inanimate subjects, nor between abstract and concrete concepts etc. Without this basic knowledge, however, any other assertions

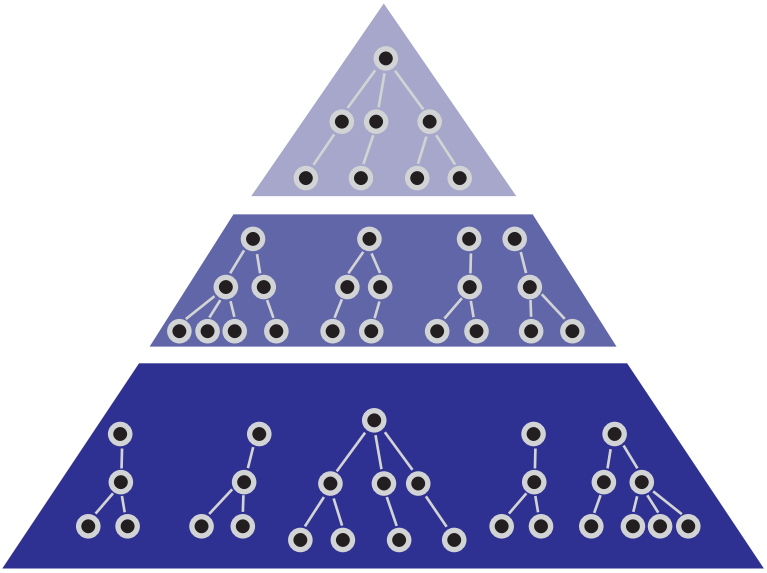

Figure 2.3: Upper, middle and lower ontology partitionings [MNV02].

lack proper foundations.
As a result from these observations, ontologies are usually partitioned into *upper*, *middle* and *lower ontologies*. Upper ontologies only contain approximatley a dozen concepts, like "(in)tangible subject", "space", "time", "event", "individual", "set" etc. These concepts are considered to be domain independent and are therefore valid for almost any possible ontology. Middle ontologies contain a few hundred concepts that are mostly domain independent, but may not be required by a certain ontology; they include concepts like materials, agents, weather conditions, plants and animals etc. Lower ontologies finally are domain specific ontologies that only conver their respective area of interest, like a certain branch of science. Typically, these domain specific ontologies contain many thousands of concepts and relations between them [MNV02]. This partitioning of ontologies is shown in Figure 2.3.
The distinction between upper, middle and lower ontologies however is not as precise as depicted above. In many papers, ontologies are considered "upper" ontologies if they contain knowledge that represents human common knowledge up to a varying degree of detail. Most so-called "upper ontologies" are therefore actually some combination of upper and middle ontologies with respect to the concepts they contain:

> "Upper-level ontologies capture mostly concepts that are basic for human understanding of the world. They are grounded in [...] the common sense that makes it hard to formalize a strict definition for them. They represent the so called prototypical knowledge." [KSD01]

Since upper and middle ontologies are "limited to concepts that are meta, generic, abstract and philosophical, and therefore are general enough to address (at a high level) a broad range of domain areas" [SUO], they necessarily contain almost the same concepts or at least concepts that are closely related to each other. This lead to the creation of generally available upper ontologies that can be used by any ontology author to build his ontology on top of. Currently, the best known upper ontologies are Cycorp's *OpenCyc Ontology* and the *Suggested Upper Merged Ontology* (SUMO), which both are proposed starter documents for the creation of a free, public standard called "Standard Upper Ontology", or SUO.
Of course, the creation of upper ontologies also poses some serious, mostly philosophical questions: there is for instance an endless debate on whether physical objects "are completely present at any moment of their existence" or should be regarded not different from processes (in time)[6]. Such discussions can potentially lead to the creation of different upper ontologies that are incompatible to each other due to their fundamentally differing views of the universe. A possible workaround has been proposed by Chris Menzel, Associate Professor of Philosophy at Texas A&M University and involved with the creation of the SUMO, which is to bundle up the various representational choices in consistent and independent packages and create a lattice from these:

> "The top node of this lattice would be the SUMO, and each level below the top node would represent inconsistent formal theories that could be used in conjunction with the SUMO. Thus, each path through the lattice from the top node to a lowerlevel node would result in a formal theory that is self-consistent, but inconsistent with various other representational choices provided by the ontological lattice." [NP01]

The philosophical problems mentioned above are also recognized by Atanas K. Kiyakov et al.:

> "The existence of several upper-level ontologies that disagree on the most basic concepts about the entities in the world demonstrates a significant philosophical diversity. [...] Which properties of the entities in the world are the most basic ones? What follows from different choices on this level? On which level of generality the differences disappear if they disappear at all?" [KSD01]

Similarly, Missikoff et al. also point out the conceptual difficulties which are encountered upon creation of an upper ontology:

[6] The respective notions are also known as "3D" (the "endurantist") or "4D" (the "perdurantist") views.

> "Creating ontologies is a difficult process that involves specialists from several fields. Philosophical ontologists and Artificial Intelligence logicist are usually involved in the task of defining the basic kinds and structures of concepts [...] that are applicable in every possible domain. The issue of identifying these very few 'basic' principles [...] is not a purely philosophical one, since there is a clear practical need of a model which has as much generality as possible, to ensure reusability across different domains." [MNV02]

Creating an upper ontology is therefore far from being a straigh-forward task, which makes it difficult for a single person or even for a specific branch of science to create an upper ontology that is actually useful for anybody else.

> "Research in computer science, artificial intelligence, philosophy, library science, and linguistics are helping to meet the need for a comprehensive, formal ontology. All of these fields have experience with creating standard descriptions and terminology for the entities and events that make up our world. However, none of these fields has been able, on its own, to construct a standard, upper-level ontology. Computer scientists and philosophers lack consensus in their communities for creating the very large, wide-coverage ontologies that are needed, although they have the necessary formal languages to do so. Librarians and linguists have the charter to create large ontologies, but those ontologies have typically lacked the formal definitions needed for reasoning and decision-making." [NP01]

The following sections will shortly introduce the two largest upper ontologies available today, OpenCyc and SUMO.

### Cyc and OpenCyc Ontologies

In 1984 the Stanford professor Douglas Lenat startet the Cyc project, named after the word "encyclopedia", with the goal to establish the world's first knowledge base for human common sense. In 1994, Cycorp, Inc. was founded to further drive the initial objective. The official vision of Cycorp is "to create the world's first true artificial intelligence, having both common sense and the ability to reason with it" [Cyc]. At present, the Cyc knowledge base consists of a very large knowledge base containing about 100,000 terms (see Figure 2.4 for an exemplary selection) and 12,000,000 assertions that are organized in thousands locally-consistent contexts, the so called "microtheories".
In addition to the knowledge base, the Cyc system contains a natural language processing module (NLP) for the English language and, first of all, a powerful inference engine that is optimized in many ways to deliver fast answers to user queries. According to Cyc, the inference engine "performs general logical deduction (including modus

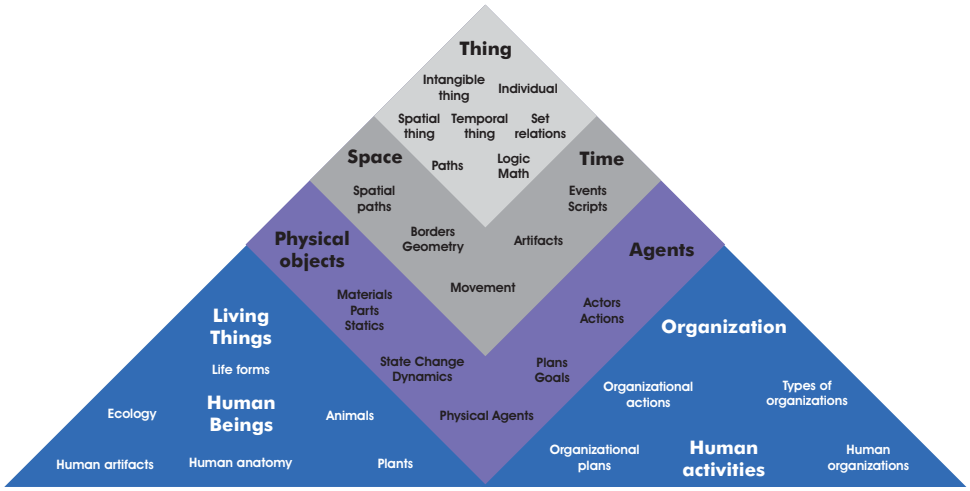


Figure 2.4: Incomplete upper part of the Cyc Knowledge Base [Cyc].

ponens, modus tolens, and universal and existential quantification), with AI's well-known named inference mechanisms (inheritance, automatic classification, etc.) as special cases." It also uses a set of heuristics to search over proof-space, which is necessary for searching a knowledge base that large: "many approaches commonly taken by other inference engines (such as frame-based expert system shells, RETE match, Prolog, etc.) just don't scale up to KBs of this size". The inference engine does not rely on the "closed world assumption" commonly found in traditional database systems, i.e. that the non-existence of an assertion automatically infers the existence of the corresponding negation. It should also be mentioned that the Cyc inference engine supports "default-reasoning", see Section 5.4.

All facts in the knowledge base are expressed in a language named "CycL", which is "essentially an augmentation of first-order predicate calculus (FOPC), with extensions to handle equality, default reasoning, skolemization, and some second-order features."[7] The CycL language is rather easy to learn and due to its extensions of first-order logic closer to the expressiveness of human language than other logic-based languages. One drawback is the consequence that there exist assertions that are undecidable, which may be problematic with respect to computational efficiency.

One important feature of Cyc's knowledge base is that all assertions are organized in contexts, or *microtheories*. These microtheories each contain a number of assertions that are based on a shared set of assumptions, a common topic and a shared source. The truth of the assertions depends on the assumptions in the microtheory. Microtheories are organized in a polyhierarchy with the effect that assertions, that are true in

[7] Taken from `http://www.cyc.com/cyc/technology/whatiscyc_dir/howdoescycreason`.

one microtheory, are also true in any microtheory below in the hierarchy. Assertions within microtheories that are not related in such a way may be contradictory, which for instance allows for formulating diverging views upon the truth of certain facts. "The contexts are first-class terms in the KB; they appear in assertions, they are organized into an ontology, and so on. An mt's assumptions are themselves Cyc assertions about time, space, granularity, topic, etc." [RL02].
However, the Cyc ontology is a proprietary development of Cycorp and therefore in its entirety not publicly available. In 1997, Cycorp released the *Cyc Upper Ontology*, a text file containing the most general concepts defined in the ontology so far, but without any rules and microtheories. In 2002, Cycorp released an extract of its ontology called *OpenCyc* to the public (under the GNU Lesser General Public License) containing most upper level concepts and some mid-level ones. Release 1.0 of OpenCyc will contain about 6,000 concepts and 60,000 assertions. Together with the OpenCyc knowledge base, Cycorp also offers tools for querying, inferencing and adding facts to the knowledge base.

### Suggested Upper Merged Ontology

The *Suggested Upper Merged Ontology*, or SUMO, "is an upper level ontology that has been proposed as a starter document for The Standard Upper Ontology Working Group, an IEEE-sanctioned working group of collaborators from the fields of engineering, philosophy, and information science. The SUMO provides definitions for general-purpose terms and acts as a foundation for more specific domain ontologies" [NP01]. In the source file[8] for the SUMO it is stated that the SUMO "was developed within the SUO Working Group by merging the SUO "candidate content" sources and refining and extending this content on the basis of various knowledge engineering projects and input from the SUO Working Group":

> "The SUMO incorporates elements of John Sowa's upper ontology (as described at `http://www.bestweb.net/~sowa/ontology/toplevel.htm` and in Chapter 2 of his book "Knowledge Representation", Brooks/Cole, 2000), Russell and Norvig's ontology, PSL (Process Specification Language), Casati and Varzi's theory of holes, Allen's temporal axioms, the relatively noncontroversial elements of Smith's and Guarino's respective mereotopologies, the KIF[9] formalization of the CPR (Core Plan Representation), the ontologies available on the Ontolingua server maintained by Stanford University's Knowledge Systems Laboratory, the ontologies developed by ITBM-CNR, some of the spatial relations from an unpublished

[8] Version 1.6 can be downloaded from `http://einstein.teknowledge.com:8080/download/register.jsp?fileType=.zip&fileName=Sumo.zip` after registration
[9] Knowledge Interchange Format, see Section 1.1.3.

**Physical**
- **Object**
  - **SelfconnectedObject**
    - **ContinuousObject**
    - **CorpuscularObject**
  - **Collection**
- **Process**

**Abstract**
- **SetClass**
  - **Relation**
- **Proposition**
- **Quantity**
  - **Number**
  - **PhysicalQuantity**
- **Attribute**

Figure 2.5: The SUMO top level ontology [NP01].

> paper by Iris Tommelein and Anil Gupta entitled "Conceptual Structures for Spatial Reasoning", and a "Structural Ontology" proposed by David Whitten and substantially revised and extended by Chris Menzel."

The SUMO is copyrighted by Teknowledge and released under the GNU General Public License. Figure 2.5 shows the top level concepts of the Suggested Upper Merged Ontology.
The purposes of the SUO project include automated reasoning to support knowledge-based reasoning applications, interoperability between applications (by establishing a neutral standard that another ontology can be mapped to) and providing a basis for e-commerce applications and educational applications as well as natural language understanding tasks.
A somewhat simplified version of KIF called SUO-KIF [KIF] is used as serialization format for the SUMO ontology. Similar to CycL, KIF has declarative semantics and is logically comprehensive (it allows the expression of arbitrary sentences in the first-order predicate calculus).
With respect to the number of concepts represented, the SUMO is somewhat smaller than OpenCyc and focuses more on real "upper level" concepts than on concepts that are supposed to be in the middle level of an ontology. Currently, SUMO contains about 1,000 concepts, 4,000 axioms and about 800 rules. While recognizing the impressive size of the Cyc ontology and the amount of work put into its creation, the people involved with the development of the SUMO also enumerate some limitations of the

Cyc ontology and respective advantages of the SUMO:

- only a small part of the entire Cyc ontology (namely the OpenCyc ontology) has been released to the public;
- Cycorp retains proprietary rights on its ontology, which is directly opposed to a potential use as public standard, whereas the SUMO "is the working paper of an IEEE-sponsored open-source standards effort"[10];
- the contents of the Cyc ontology have not been reviewed extensively by independent experts, while the content of the SUMO was not only created by a "diverse group of collaborators from the fields of engineering, philosophy, and information science", but also heavily influenced by an open discussion on the SUMO mailing list;
- the SUMO "should be simpler to use than Cyc" as "any distinctions of strictly philosophical interest have been removed" [NP01].

For a critical discussion of the usefulness of (standard) ontologies, see Section 5.3.

## 2.5 Ontologies and Schemata

Since the early days of computer-supported, automated data processing, *relational database management systems (RDBMS)* have been used to organize and store data of various kinds. Over the years, many optimizations such as indices etc. as well as "secondary functionality" (e.g. security mechanisms) were added to many systems, and RDBMS evolved into highly efficient backend storage facilities used by innumerable applications. The query and data manipulation language SQL also contributed to the popularity of relational databases. With the advent of object-oriented programming languages, the relational design of databases was questioned and alternatives were developed, but were not adopted widely, mainly due to their inferior performance figures compared with the "old-style" relational databases.
While relational databases proved to be well-suited for a wide range of different application scenarios, their biggest strength turned out to be also their greatest weakness: the fixed schema of a database allows for efficient organization of data, but is inappropriate whenever changes to it are expected to occurr often. This was especially recognized as hindrance for knowledge management systems, as they often require a certain flexibility of data structures. Therefore, different organization methods for

[10] Actually, both the OpenCyc ontology and the SUMO are starter documents for the SUO; however, the advocates of the SUMO seem to have greater influence on (and are maybe more interested in) the SUO working group than the Cyc creators do.

data are needed for such systems, and ontologies offer interesting capabilities in this area.
One of the big advantages of ontologies over traditional (object-)relational databases is indeed their property to actively support the evolution and necessary change of knowledge, i.e. knowledge maintenance. This is mainly due to the fact that ontologies do not have to obey a fixed schema[11], as database systems do; this will be explained in the next paragraphs.
Traditionally, database systems use tables and similar structures, e.g. in the case of object-oriented databases, to specify a schema for the actual data to be filled in. The layout of the tables is deferred from the concepts that are to be stored within them. While the data (i.e. instances of the concepts) can easily be added, modified, deleted and of course queried, this generally not applies to the information contained implicitly in the definition of the schema. Although it is possible to retrieve information on the layout of the tables (often called *metadata*) e.g. by querying a special "master table", changes to the schema, while definitely possible, are problematic in a number of ways. Since the information contained in the schema is only of implicit nature, it is usually also transferred implicitly to the application that connects to the database *at the time of development*. This means that during the creation of the application, information about the schema (and hence the layout of tables and other objects in the database) is encoded within SQL-statements, APIs and even elements of the graphical user interface (GUI). Any changes to the schema must therefore be reflected by every application that accesses the data described by that schema, which is the reason for the static layout of databases: generally, a database schema is created at the very beginning of the software development cycle, possibly modified later on, but surely not changed any more after the release of the application when production data is stored by the customer.
A fixed schema has the undeniable advantage that it allows the data to be organized very efficiently, for instance through the creation of supporting structures like indices, compiled stored procedures etc. Also, programmatic data structures (e.g. Java classes etc.) and even many elements of the user interface often resemble the underlying structure of the database schema. But the advantage of efficient organization comes at the cost of flexibility: in fact, the need for *frequent structural changes* is completely ignored, with the term "frequent" meaning "occurring too often to adapt application code manually". This however is exactly what knowledge management (and also novel approaches such as the view of corporations as "adaptive enterprises") is about: only by constantly adding and modifying *concepts* in the knowledge base, effective knowledge maintenance takes place; a fixed schema is rather inacceptable for a on-

[11] The existence of "schema languages" like RDFS does not contradict the fact that ontological schemas are not fixed; rather, schema languages allow for creating ontologies that *represent* a certain schema but are not subject to its constraints *in their entirety*. An explanation of RDFS can be found in Section 3.2.2.

tological knowledge base. The implications of these observation are far-reaching; for instance, the traditional roles of the *schema designer* (i.e. the database designer) and the application user become indistinguishable or at least overlap.
One could of course say that the classes in an ontology effectively make up some kind of *flexible schema*, like a "virtual layer"[12] side by side with the set of instances of classes (which could be seen as a second "layer", accordingly). This is also what Hans Holger Rath refers to by stating that in order to get a Topic Map application up and running, it is essential to "define the schema – what kinds of subject will be covered and how they will be related; what should a valid and consistent topic map look like" [Rat03]. Similar statements can be found also for RDF: "We can distinguish three kinds of concepts in RDF: fundamental concepts, schema-definition concepts (useful for defining new vocabularies) and utility concepts [...]" [Cha01]. While this point of view is certainly not incorrect, it obscures the great potential that is inherent to any application using an ontological structuring of data. This is even more true for large scale ontologies: "... we know that building large scale ontologies pose a new set of problems especially in an environment where ontologies are viewed as 'live repositories' rather than frozen resources" [Var02].
Actually, for ontology applications it is far more appropriate to completely abandon the concept of a schema: since there is no implicit schema in an ontology, why should one try to artificially create one? This approach is also endorsed by the fact that the distinction of classes and instances is not as natural as it might appear at first. As already outlined in Section 2.4, any concept can be a class and an instance *at the same time*, which makes a clear separation between the two impossible. Both Topic Maps and OWL Full support the representation of a concept as both class and instance; OWL Lite and OWL DL do not allow such kind of representation, but require an (artificial) distinction and according workarounds for computational reasons. In case of Topic Maps, it is even impossible under certain circumstances to specify if a given subject is an instance or a class, or both (this is known as the "sleeping class" problem, see also Section 3.1.4).
Instead of creating a schema, the focus lies on the *constraints* that may be applied to instances of classes, and which may be inherited by subclasses. Such constraints are usually established either through the ontology representation format itself, as it is the case with RDF/OWL, or they may be superimposed by an additional constraint language in case of Topic Maps. It should not be neglected the fact that the absence of a fixed schema has severe implications on storage considerations and the design of applications (and especially their user interfaces), which are beyond the scope of this thesis.
Finally, the role that (object-)relational databases can play in the field of ontologies should be analyzed more thoroughly. In fact, most ontologies rely on relational

[12] as opposed to the static layer which a database schema represents

databases for persisting their data, which looks like a contrast to what has been said above. As explained in Section 4.1, the key difference is the mapping between concepts and schema elements (tables, ...) of the database. Originally, concepts of the *application domain* were mapped to the database schema: e.g. a table "EMPLOYEE" for the concept (or class) of employees, a table "CLIENT" for the class of clients etc. Since ontologies do not know the number and definition of the application concepts beforehand, they instead map *their own ontological language constructs* to the database schema: e.g. a table for all "classes", a table for all binary "relations" and so on. Used in this way, traditional databases still play a crucial role for persistency and scalability of ontologies.

# 3 Topic Maps and RDF/OWL

This chapter deals with the details of the ontology representation formats Topic Maps and RDF/OWL with respect to their functionalities as identity-based technologies. The first section starts with a short history and current status of Topic Map standardization and continues with an in-depth explanation of concepts like *subjects*, *topics* and *assertions* that are crucial for understanding how Topic Maps work. Further sections will introduce the concepts of *class hierarchies*, *scopes* and *reification* of subjects or assertions, as well as the ideas behind *published subjects*.
The second section covers RDF, RDFS and OWL and their respective language concepts. Special attention is further given to the relationship between Topic Maps and RDF by comparing the two technologies at structural level, correlating the corresponding standards and pointing out similarities and differences.

## 3.1 Topic Maps and Related Standards

### 3.1.1 History of Topic Map Standardization

In 1991, the Davenport Group was established with the goal to create a standard for software documentation [Rat03]. One of the main contributors was the publisher O'Reilly & Associaties, which needed a standardized master index for the X-Windows documentation. Two subgroups were formed, one developing the DocBook DTD, the other, named "Conventions for the Application of HyTime" (CApH), creating the SOFABED (Standard Open Formal Architecture for Browsable Electronic Documents) model which used the hyperlink facilities provided by the ISO 10744 HyTime standard. The CApH group elaborated the SOFABED model and renamed it "Topic Maps"; in 1995, it was accepted as "new work item" by ISO and released in 1999 by the ISO/IEC JT1 SC34 as International Standard ISO/IEC 13250 [BBN02].
The ISO/IEC 13250 standard defined an interchange syntax based on SGML and HyTime (HyTM, for HyTime Topic Maps), which according to the "Guide to topic map standardization" [BNB02] was not compliant to resources on the World Wide Web (e.g., HyTime does not necessarily use URIs to reference external resources). To resolve this issue, the independent organization Topicmaps.org was established in 2000 with the goal to elaborate a Topic Map specification based on the W3C recommendations XML and XLink. In 2001, the official version of the XML Topic Map (XTM) 1.0 specification was published. This specification was given to ISO/IEC, which included

the XTM notation as second interchange syntax in the original 13250 standard by means of a Technical Corrigenda in October 2001.
Although both interchange syntaxes HyTM and XTM are specified in ISO/IEC 13250, no explanation or definition is given on how the two syntaxes relate to one another. Also, both specifications fail to define precise guidelines for implementations, resulting in a number of situations where it is unclear what exactly is supposed to happen within a Topic Map application. Implementations are forced to define their own rules for such cases sacrificing interoperability, which is quite contrary to the original purpose of an International Standard. Differences between the two interchange syntaxes are described in [BN01].
To overcome such situations, a new edition of the ISO/IEC 13250 standard will be created. The core part of the new standard will be formed by the Topic Map Data Model (TMDM, formerly known as Standard Application Model, SAM), which will provide a precise and unambiguous basis for Topic Map implementations. Another important element of the new standard will be the Topic Map Reference Model (TMRM), which is a more abstract graph model for Topic Maps and will feature "mechanisms for explaining relationships between different knowledge representations" [BNB02].
In 2001, two more standard initiatives were started by the ISO/IEC technical committee JT1 SC34: the Topic Map Query Language (TMQL), which will become International Standard ISO/IEC 18048, and the Topic Map Constraint Language (TMCL, ISO/IEC 19756). Both specifications rely on the TMDM and are still under development as of January 2005; for details about the existing drafts, see sections 4.2.2 and 3.1.10 respectively.

All parts of this thesis which deal with Topic Map syntax representation, especially those concerning storage and querying of Topic Maps, will always refer only to the XML Topic Map (XTM) syntax (general concepts and considerations, on the other hand, are usually independent from the serializing syntax and therefore valid for all notations). The reason for this is that the underlying XML notation has become extremely popular during the last years, whereas only a marginal number of HyTime applications exist. Additionally, as already mentioned earlier, the addressability of electronic resources using URIs is an important requirement for Topic Maps, considering the broad availability of the World Wide Web or HTTP-based intranets. Future Topic Map implementations will therefore almost certainly use the XTM syntax for representing Topic Maps.

### 3.1.2 Topic Map Notations

Before discussing any details of Topic Maps, a short introduction on the different Topic Map syntaxes is given here. The Topic Map standard ISO 13250 determines general conditions that must be satisfied by all (standardized) Topic Maps. The most

important ones regarding the notation of Topic Maps will be quoted here:

> "The Topic Map notation is defined as an SGML Architecture, and this International Standard takes the form of an architecture definition document expressed in conformance with Normative Annex A.3 of ISO/IEC 10744:1997, the SGML Architectural Form Definition Requirements (AFDR). The formal definition of the Topic Map notation is expressed as a meta-DTD.
>
> [...]
>
> The Topic Maps syntax makes use of the base, location address, and hyperlinking modules of the HyTime architecture as defined in clauses 6, 7 and 8 of ISO/IEC 10744:1997.
>
> [...]
>
> The HyTime architecture provides a comprehensive set of addressing mechanisms and a standard syntax for using them. In addition, it provides means whereby any addressing syntax can be declared and used. The Topic Map architecture preserves these features of HyTime. Thus, the Topic Maps architecture allows Topic Map authors to use any addressing scheme, including proprietary addressing mechanisms driven by expressions in any notations, provided each such notation is formally declared as a notation in the manner prescribed by the SGML and HyTime International Standards." [BBN02]

One very important thing to note is that the original Topic Map notation is using SGML and HyTime (which are both ISO standards) to define a *meta-DTD*. This meta-DTD can be used either directly to create Topic Map documents, or indirectly to create different Topic Map notations (which derive from the meta-DTD and thus are still compliant to the original standard).
This makes it possible to create different "flavors" of topic map notations which is not only advantageous but also adds significant complexity to real world Topic Map applications. In order to overcome this situation, the standard was extended to include the XML Topic Map (XTM) notation, which is based on XML and XLink. In fact, XML is a subset of SGML that covers the most important aspects of the very complex and lengthy SGML standard, but is easier to handle.
Both the HyTM and XTM notations are defined in the Topic Map standard and are therefore in the status of International Standards. Apart from these, a number of proprietary notations exists that either were created when no standard had been defined yet or that have advantages regarding the verbosity or complexity of the standardized notations. The most well-known notations are the *Linear Topic Map Notation* (*LTM*, [Gar02]) defined by Ontopia AS, and AsTMa= (which is part of the AsTMa*

language family, [Bar04]). Both notations are more compact than the XTM notation, but while LTM is intended as interchange format between machines (as is XTM), AsTMa= is far easier to read and write directly by humans.
All notations have in common that they are always used for *serialization purposes* only, for instance to write topic maps to physical files or to exchange Topic Maps between machines. They are not relevant for the internal representation of a Topic Map within a Topic Map engine; rather, any Topic Map engine should be able to serialize its Topic Map representation using an arbitrary notation. The Topic Map Data Model states:

> "Topic maps may be represented in many ways: using topic map syntaxes in files, inside databases, as internal data structures in running programs, and even mentally in the minds of humans. All these forms are different ways of representing the same abstract structure." [GM05]

Part 4 of the revised ISO 13250 standard will also include a notation for *canonicalization purposes*:

> "Canonicalization is the process of serializing a data structure in such a way that two data structures considered to be the same result in the same serialization and two data structures not considered to be the same result in two different serializations. A canonical form enables direct comparison of two data model instances to determine equality by comparison of their canonical serialization.
>
> This part of ISO/IEC 13250 defines a canonical sort order for any set of of information items from the Topic Maps Data Model and a transformation of an instance of the Topic Maps Data Model to an instance of the XML Infoset model." [Ahm04]

Thus, the canonical serialized form of two Topic Maps (or their constructs) can be used to determine their equality or inequality by comparing them byte-by-byte.

### 3.1.3 Topics – Proxies for Subjects

Within Topic Maps, the proxies for representing subjects are called *topics*; one may also say, a topic is the electronic representation for a subject which is otherwise not addressable or identifiable by a computer. However, as mentioned in Section 2.1, this statement is too restrictive: actually, Topic Maps treat everything the same way, whether it is a resource that already has an address, or anything else: *every* subject is represented by a topic. There is even a 1-to-1 relationship between subjects and topics: in a Topic Map, a subject is represented by exactly one topic, and one topic always represents exactly one subject.
For electronic resources that already have an address which identifies them uniquely, a topic may use that address as *subject address*. If two topics have the same subject

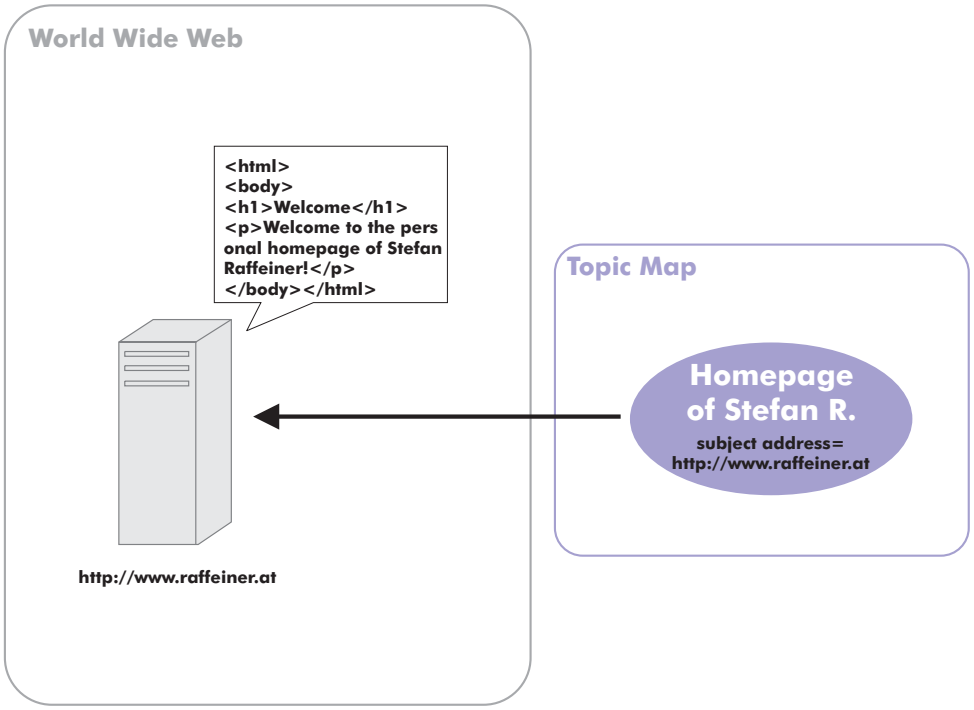


Figure 3.1: Subject representation with subject address.

address, they are considered to represent the same subject (and, as there can be only one topic representing a subject in a single Topic Map, the two topics would be merged). Figure 3.1 shows an example of a Topic Map containing a single topic that represents the author's homepage: as the subject (the homepage) in this case is clearly a resource with an address, the topic uses this address as *subject address* for the purpose of identification.
All subjects that are not network-retrievable resources are represented by arbitrary electronic resources which give some idications on the identities of those subjectes. Such a resource is called *subject indicator*, its address is the *subject identifier*. Both the subject address and the subject indentifiers are basically computer addresses (e.g. URIs), but they are used for different purposes. Subject addresses, indicators and identifiers are explained in Section 2.2.
Figure 3.2 shows an example of a Topic Map containing a single topic that represents a person: as the subject (the person) has no network address, the resource

`http://www.raffeiner.at`

is used as subject indicator. The address of the subject indicator is the subject identifier used by the topic to reference to the author.

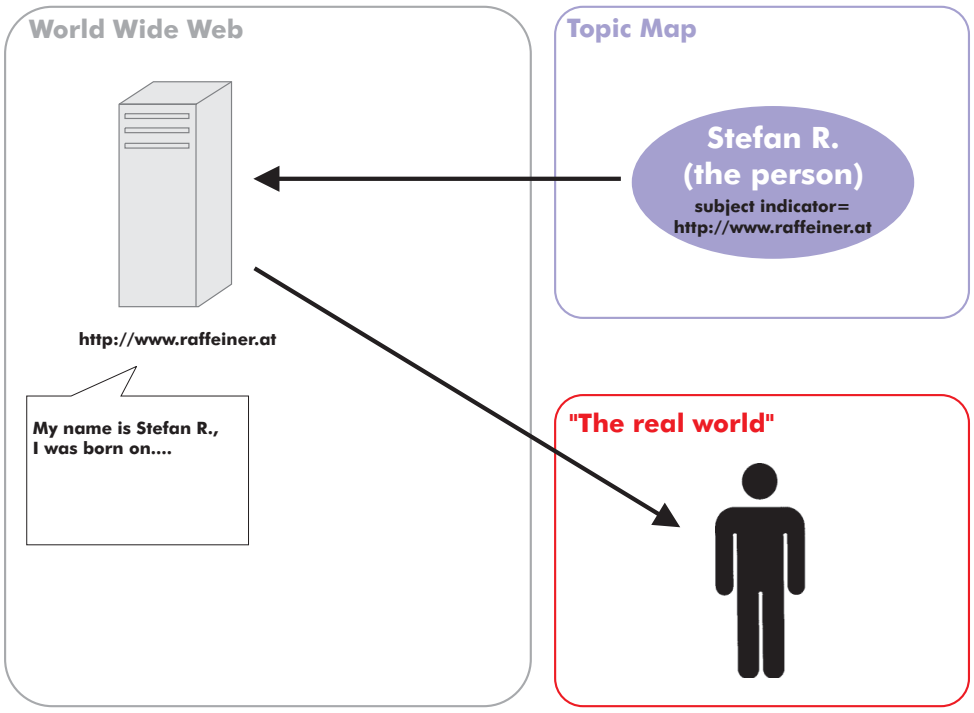


Figure 3.2: Subject representation with subject indicator and subject identifier.

If a subject is likely to be represented in more than one topic map, it is advisable not to use an arbitrary subject indicator but refer to *published subjects* instead; see Section 2.3 for details about published subjects and published subject indicators.
The decision whether a certain subject (or, topic) should be included in a Topic Map is to be made by the Topic Map author(s) with respect to the purpose or domain of the Topic Map. From a conceptual point of view, any topic of any type may be included in a Topic Map.

### 3.1.4 Classes and Instances

Topic maps provide a very elegant way to deal with classes and instances: as pointed out in Sections 2.2 and 2.1 about subjects and identification, in a Topic Map *all* subjects are represented by topics, including abstract ones. This means that there is no substantial difference between instances and classes of subjects in a Topic Map: every instance and every class gets a topic which represents it. There is in fact no way to tell whether a topic is a class or an instance unless there exists another topic that states that it is an instance of the first topic (which must be a class then); this is also known as the "sleeping class problem" (introduced in the Topic Map Handbook [Rat03]) and should be approached with the development of the TMCL. All topics that do not have an explicitly defined "instance of" relationship are implicitly instances of the topmost class "topic", which is predefined in the Topic Map standard.
Regarding the various elements of a Topic Map, it can be stated that:

- a *topic* can be instance of multiple classes
- an *occurrence* can be instance of exactly one class
- an *association* can be instance of exactly one class
- no other elements can be instances of classes.

### 3.1.5 Topic Characteristics: Names, Associations and Occurrences

Since topics provide references to subjects by means of subject addresses or subject identifiers, any information that should be tied to a subject can be stored in the Topic Map and then bound to the representing topic; the containers for storing information about subjects are called *topic characteristics*. There are three different kinds of topic characteristics: names, occurrences and associations. The following sections will provide some insight into the details of each.

#### Names

The simplest and often most important information about a subject is its name, which is usually basically a string. Without a name, a topic is only represented by an abstract

id, which is not suitable for communication with humans. A topic can have multiple names, e.g. to list synonyms or translations in different languages or to enable accurate sorting. Topic Maps know two different types of names: *base names* and *variant names*. A base name is a string primarily used for identification by humans; it is also used for display and sorting if no variant name exists that is specifically suited for these tasks. Variant names are considered to be "variants" of the base name, e.g. synonyms or translations etc.
Base names are governed by the *Topic Naming Constraint(TNC)*: the XML Topic Map notation standard states that:

> "Two topics with the same base name in the same scope[1] implicitly reify the same subject and should therefore be merged." [Gro01]

The TNC has always been a somewhat controversial element of the Topic Map standard and, according to various mails on the Topic Map mailinglist[2], is very likely to be made an optional feature in the final version of the Topic Map Data Model.

### Occurrences

With exception of names, all facts about a single topic are represented by *occurrences*. There are two different ways to use occurrences: if an external resource like a web page or a document exists, the occurrence is basically a reference to that resource (e.g. this can be used to connect a picture to the topic representing a certain person). If some kind of "simple" data should be linked to a topic (like "20.10.1980" or "123,000"), the data itself is stored directly within the occurrence element (and therefore within the Topic Map). The Topic Map standard does not define any data types; all occurrences are stored as strings and must be interpreted by the application (which also has some effects on efficient storage of Topic Maps, see Section 4.1).
In order to express the "meaning" of an occurrence it is possible to assign a certain type, or class, to an occurrence by stating that the occurrence is an instance of that class. For example, the occurrence referencing

`http://www.raffeiner.at/pic1.jpg`

may be instance of the class "portrait", and the occurrence "20.10.1980" may be instance of the class "birthdate".

### Associations

Establishing relations between topics in a Topic Map is done by creating *associations*. Associations can include any number of topics and are therefore said to be "n-ary"; the

[1] For details on scopes, see Section 3.1.6.
[2] see for instance `http://www.infoloom.com/pipermail/topicmapmail/2004q3/006221.html`


Figure 3.3: An exemplary association "is capital of".

most commonly used associations are binary (i.e. connecting 2 topics). Associations are assertions and have therefore *no inherent direction*; instead, every association specifies *roles*, and the topics to be connected through that association are said to play these roles (*role players*). This implies that an association can be seen from the perspective of any participating topic and will still represent the same assertion: it is the same thing to say "Vienna is the capital of Austria" and "Austria has a capital named Vienna". Figure 3.3 shows a Topic Map with two topics, Austria and Vienna, and the association "is capital of" between them. The association has two roles, "country" and "city", which are played by the topics Austria and Vienna, respectively.
It should be pointed out that the incorrect assertion "Austria is the capital of Vienna" can only be achieved by assigning the wrong roles to the role players (Austria plays the role of city and Vienna the role of country), which must be detected and prevented by some semantic validation mechanism, which could be provided for instance by the TMCL. The Topic Map standard itself has offers no vocabulary for specifying constraints on which classes can play which roles. As it is the case with occurrences, also associations can be instances of classes which determine the "meaning" of an association and also define the roles which exist for that association.

### 3.1.6 Scopes

A scope represents a certain point of view on the information in a Topic Map, which means that the validity of assertions can be limited to certain circumstances. Scopes are ideal for representing languages: the validity of the base name "Vienna" of a topic could be limited to the scope "English", whereas the base name "Wien" for the same topic is only valid for the scope "German". Other potential applications for scopes include the propagation of access rights or the expression of different or contrary opinions about characteristics, which is an important feature especially for knowledge management. Representing contrary opinions is however difficult to implement with scopes, as outlined in Section 5.2.
Scopes can be specified for all three topic characteristics: base names, occurrences and associations; these characteristics are then said to be *scoped*. The topics that the characteristics belong to are *not* part of a scope, but exist independently from scopes. If no scope is defined for a topic characteristic, the default *unconstrained scope* applies.

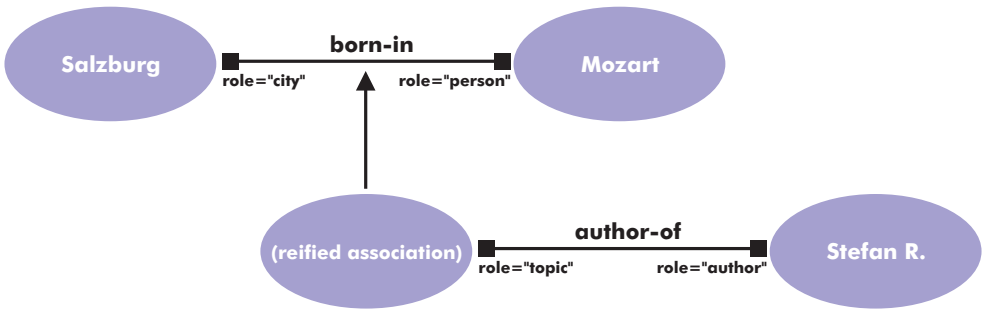


Figure 3.4: Reification of an association.

The unconstrained scope does not limit the validity of the assertion expressed by the characteristic.
A scope can effectively be used to *filter* a Topic Map: all characteristics that belong to a scope can be "hidden" or "shown", depending on the visibility of that scope (within a user interface, for instance). A scope can therefore also be seen as a set containing all characteristics that belong to it.
Although scopes are a very important feature of Topic Maps, it is "a feature of which semantics have been deliberately left fuzzy by the Topic Maps specification, letting to every application the task to interpret it at will in its context." [Vat03]. This can get problematic if Topic Maps are to be used in a less controlled environment or are supposed to be merged (which was originally one of the motivations for creating the Topic Map standard).

### 3.1.7 Reification of Assertions

Topic Maps provide a feature which allows Topic Map authors to make *statements about statements*:

> "The concept of *reification* has had the misfortune to have had stuck onto it a rather intimidating name, but it is in fact quite simple. Making assertions about things is easy, as you can create symbols[3] to do so and then just make the assertions. Making assertions about assertions, on the other hand, is tricky. However, if you can create a symbol that represents an assertion you can just use that symbol to do it. Creating a symbol that represents an assertion is exactly what reification is." [Gar03a]

For example, a small Topic Map could consist of the topics "Mozart" and "Salzburg" and a relationship of type "born in" (Figure 3.4). If the Topic Map should include a reference to the author of that relationship, this means to make a statement about

[3]The term "symbol" is used for "topic" here.

the relationship. In order to do so, the relationship has to be reified first: in order to do so, a new topic is added to the Topic Map and its subject indicator is set to the association between "Mozart" and "Salzburg". The new topic now represents the the association itself; arbitrary new associations (and occurrences) can now be added to this topic, e.g. an association of type "author-of" to specify the author of that association.
As depicted in the example, reification of Topic Maps is basically performed by simply adding a new topic to the Topic Map. As aleady indicated by Figure 3.4, reification in Topic Maps is that easy because the relationship between the topics is already represented by an explicit "association" construct *before any reification actually takes place.* One could also say, that Topic Maps in fact reify all associations *by default* through the association constructs provided by the Topic Map standard. This "reification by default" is also called *preemptive reification*; its advantages are described by Steven R. Newcomb as follows:

> "The advantage of preemptive reification is that we can always say something new about anything that already exists, without having to choose between the two evils of creating redundant connections, on the one hand, or invalidating existing lore about how things are connected together, on the other." [New02]

Newcomb's remarks about the "two evils" refer mainly to RDF reification, see Section 3.2.1, which is also called "lazy reification".
As Topic Maps preemptively represent all relationships through associations, there must be a limit to the depth of recursion which is implied by this behaviour. According to Newcomb, the Topic Map Reference Model forbids any "in situ"[4] reification that only serves to make assertions about Topic Map *constructs* themselves:

> "[The TMRM] uses the yardstick of 'substantiveness with respect to to the semantics of the assertion' to justify its decisions as to the level at which further *in situ* reification is forbidden. [...] *in situ* reification does not occurr when its only conceivable purpose would be to allow assertions to be made about the subterranean mechanics of the graphic representation of an assertion." [New02]

Although sometimes reification is not considered an important aspect of ontological knowledge representation, the fundamental differences between Topic Maps and RDF should be emphasized.

[4] In situ reification is the kind of reification where relationships between concepts are *replaced* with their reified versions, as it is the case with RDF reification.

### 3.1.8 Merging Topic Maps

As indicated in Section 3.1.1, one of the requirements that led to the development of the Topic Map standard was the ability to merge multiple indices into one master index. It is therefore not surprising that the Topic Map standard provides methods to join the contents of two Topic Maps.
As already mentioned in Section 3.1.3 there is a 1-to-1 relationship between topics and subjects: apart from being the logical consequence of one topic representing exactly one subject, it is also very handy for navigational purposes because all assertions (names, occurrences, associations) about a subject can be reached directly from the corresponding topic. If two Topic Maps are to be merged, it is essential to know if two topics represent the same subject: if yes, they must be merged into one topic that contains the union set of all characteristics of the original topics. There are three possibilities for two topics to represent the same subject:

- if the topics have the *same base name* in the *same scope*, the Topic Naming Constraint requires a merger[5] (see Section 3.1.5);
- if the topics use the *same subject address* to refer to an electronic resource;
- if the topics use the *same subject identifier* to refer to a subject indicator for any subject without an electronic address.

The resulting topic will be the set union of all names, occurrences and associations of the original topics (without doubles). Names are compared byte-by-byte, occurrences must be instances of the same class and point to the same resource, associations must be instances of the same class and have the same roles and role players [Rat03].
Thus, any Topic Map corresponding to the standard will never contain two topics representing the same subject. It should be added for the sake of completeness that this is not equally true for a serialized version of a Topic Map (e.g. a XML file containing a Topic Map in XTM notation). In fact, multiple topics for a single subject can be present in a serialized Topic Map, but they have to be unified according to the rules stated above upon deserialization. The purpose for this different behaviour is to facilitate easy authoring and handling of Topic Map files.

### 3.1.9 Upcoming Standards: Topic Map Data Model and Topic Map Reference Model

As explained in Section 3.1.1, the current ISO standard 13250 is about to be revised and split up in a new multipart standard. Along with a part dealing with canonicalization of Topic Maps and specifications for XML Topic Map and HyTime Topic Map

[5] The equality of subjects that have the same base name in the same scope is hard to derive logically, but should be accepted as convention rather.

syntaxes, the new standard will also contain the Topic Map Data Model (TMDM) and the Topic Map Reference Model (TMRM). Since the latter two can be seen as major cornerstones of the standard, they are briefly presented here.

#### Topic Map Data Model

As outlined in the Topic Map Data Model proposal, the TMDM "defines the abstract structure of topic maps, using the information set formalism, and to some extent their interpretation, using prose" [GM05]. The data model should facilitate the consistent interpretation of interchange syntaxes (such as XTM or HyTM as well as other, non-standardized notations) and provide a basis for other Topic Map-centric standards, such as TMCL or TMQL. The latter two however are not part of the new ISO 13250 standard. Additionally, the TMDM clarifies the rules for merging Topic Maps and defines some "fundamental published subjects" (i.e. those used to indicate the Topic Map constructs themselves).

#### Topic Map Reference Model

The Topic Map Reference Model provides a more abstract graph model of Topic Maps. It is informally described as follows:

> "In this model, names and occurrence resources turn into nodes on the same level as topics, and they are related to their topics using an association-like structure of nodes and arcs.
>
> [...]
>
> The Reference Model provides a mechanism for explaining the relationships between different knowledge representations, such as topic maps, RDF, and KIF. This will make it easier for topic maps to interoperate with these other knowledge representations." [BNB02]

If Topic Maps will ever play a major role in knowledge management, there will surely be a need for interoperability, especially if the large number of applications and ontologies using RDF and KIF is considered. The new Topic Map Reference Model supercedes earlier attempts to provide a formal model for Topic Maps, e.g. [AdMRV02].

### 3.1.10 The Ontological Potential of Topic Maps

#### Background

If one takes a close look at Topic Maps, their historical standard syntax HyTM and its XML successor XTM, and especially at their more abstract foundations (as provided by the ISO 13250 SGML standard and the TMRM), one thing stands out: there has

always been an implicit principle that guided the development of Topic Maps, and that principle still influences the most recent Topic Maps standardiziation efforts. It is the demand that Topic Maps should be able to represent "any thing whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever" [BBN02].
In fact, to get a simple Topic Map up and running, very little is needed: a few topics that represent subjects, and maybe a number of assertions to link them or occurrences that the topics can link to. There is actually no need for defining classes or types of topics since topics, associations and roles are *not required* to be typed. Associations can have any number of roles, since they are n-ary by default, and even undirected. In fact, almost no "ontological overhead" has to be added to a Topic Map, if it is not needed. This freedom of expression is certainly one of the biggest strengths of Topic Maps, as there are almost no constraints imposed through the underlying data model. It is also that principle that made the concept of (formal) ontologies rather unpopular in the Topic Map community. With respect to Topic Maps, ontologies generally use a more constraining data model with *a defined vocabulary* to represent concepts and their relationships. The ontological vocabulary allows for the creation of inference engines and the automatic semantic validation of underlying data, but on the other hand, it sometimes also limits the expressive power of the system. Such limitations have never been accepted by most Topic Map advocates. Instead, Topic Maps are preferrably referred to as being able to *represent arbitrary ontological vocabularies*. Bernard Vatant states that "both topic maps specifications and literature have implicitly or explicitly presented the standard as 'ontology-agnostic', meaning they are able to support, represent and manage any kind of knowledge in any kind of ontological context, and even independently of the constraints imposed by any ontology" [Vat04]. For instance, the Topic Map standard does not define properties of associations such as transitivity; if one needs such a property, he has to define it (e.g. by creating a topic "transitive association" and using it as type for other associations). The OWL standard, on the other hand, defines several properties such as transitivity or symmetry, but does not allow for creating arbitrary properties of relationships (and thus is said to have a fixed ontological vocabulary).
While ontologies historically clearly belonged to the field of Artificial Intelligence whose popularity has been decreasing during the last decades, the Semantic Web initiative with its W3C recommendations RDF, RDFS and especially OWL has brought ontologies to the attention of a much broader audience. Opposed to earlier ontological systems which were mainly focused on expert systems containing narrow and specialized knowledge, the Semantic Web definitively emphasizes the representation of *any kind* of knowledge, and with respect to autonomously acting agents, even human "common sense" knowledge.
This redefinition of application areas for ontologies has also confronted again Topic Maps with the claim for more ontological support. Indeed, most Topic Maps can be

found to be driven by (and thus at least partially representing) *implicit ontologies*: "Classes of topics, association types, role types, occurrence types, seem carefully chosen, and one will find out some sensible and recurrent patterns linking association types to specific role types, occurrence types to topic types, role types in association types to topic classes, implicit rules of cardinality, etc." [Vat04].
In contrast to explicit ontologies, Topic Maps provide no predefined vocabulary to impose constraints on types and cardinality, nor to facilitate the use of inference engines through relationship characteristics (like transitivity etc.) or inference rules. Although it would be possible to define such vocabularies by means of the existing Topic Map constructs, Topic Maps "enriched" in this way would become incompatible[6] with each other. In fact, the definition of constraints should be covered by the new standard TMCL, but as the existing "straw man proposal" indicates, no support for inference is added by TMCL.
The expressive power of Topic Maps is definitely not to be underestimated: Topic Maps are able to represent both a class and an instance with only a single topic, associations are n-ary and undirected (making them way more powerful than the binary, directed relationships used for instance by RDF), and they do not limit their expressiveness through any ontological vocabulary (which is not unproblematic, as explained earlier). Therefore it is not surprising that both the Upper Cyc Ontology and also the SUMO have also been exported as Topic Maps: see [Cyc01] for information about the XTM version of Upper Cyc (which itself is not available any more) and [SUM] for a browsable version of the SUMO as Topic Map. In the context of the UTON project, Topic Map constructs have also been adopted to facilitate the creation of large scale ontology networks [Var02].

#### Topic Map Constraint Language (TMCL)

The Topic Map Constraint Language is sometimes referred to as the Topic Map world counterpart of OWL, but in fact has a somewhat different focus. TMCL "provides means to express constraints on Topic Maps conforming to ISO/IEC 13250:2000" [MN03] and can thus be seen as a pure constraint language with no support for inference. Constraints expressed with TMCL can be used to guide the authoring of Topic Maps by hindering the Topic Map author to create semantically invalid assertions. Alternatively, they can be used to validate existing assertions or entire Topic Maps (e.g. prior to a merge operation).
TMCL relies on the Topic Map Query Language (TMQL, see Section 4.2.2) for specifying which Topic Map information items are subject to a specified constraint (through

[6] This means that a Topic Map application would not recognize the constraints and inference rules created by another Topic Map application, since no standardized vocabulary exists; compliance to the original Topic Map standard however would still be guaranteed.

a "selector expression") as well as for determining the validity of those information items (through a "constrainer expression").
Both TMCL and TMQL rely on the new Topic Map Data Model standard which will be part of the revised ISO 13250 standard. Since TMQL is in its early stages of development, also TMCL is far from being complete; at the time of writing, only a "straw man proposal" exists. Some people suggest to use an adapted version of OWL for Topic Map schema definitions ( [Vat03], [Gar03a], [Vat04]).

## 3.2 RDF, RDFS and OWL

### 3.2.1 The Resource Description Framework

The Resource Description Framework (RDF) is a language based on XML syntax intended for representing information about resources in the World Wide Web, or anything that can be identified by resources on the WWW. The main difference between Topic Maps and RDF is that RDF was originally designed to enable authors to *add machine-processable information* to existing network-retrieveable resources, whereas Topic Maps, although providing ways to link to resources via occurrences, represent a more abstract, index-like point of view. RDF can be considered a *bottom-up* approach that allows to make statements especially about *resources*, while Topic Maps are built *top-down*. For a discussion of the relationship between "resources" and subjects, see Section 2.1. Extensive introductions to RDF can be found for instance in [RPR04, Ogb00, Cha01].

#### Notation

For RDF, only one standardized notation exists, namely *RDF/XML*[7]. The purpose of this standardized syntax is to enable the interchange of RDF graphs between different RDF-processing applications. Thus, the rather verbous XML notation is not intended for "internal representation" (i.e. within a RDF application) of the underlying graphs. A sample RDF/XML code is shown below [RPR04]:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
```

[7] Most people referring to "RDF" as serialization format actually mean RDF/XML; not much emphasis is given to this distinction in practice.

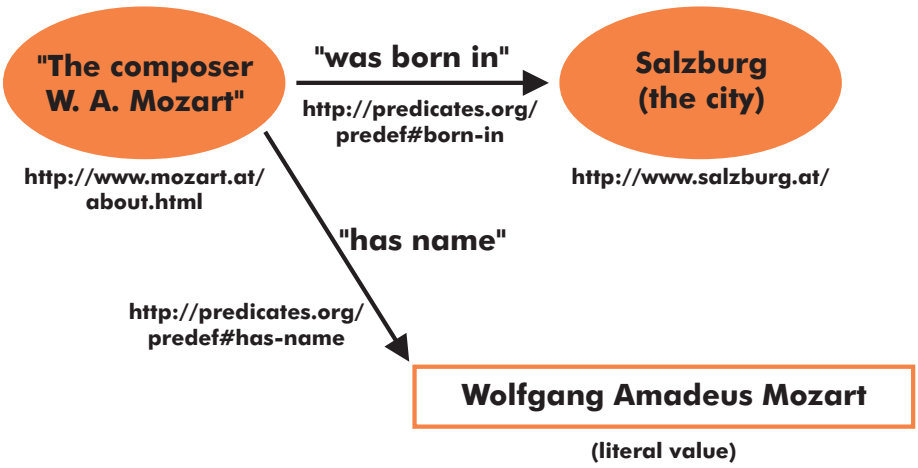


Figure 3.5: A RDF statement as graph.

```
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>
</rdf:RDF>
```

### RDF Constructs: Nodes and Arcs

The RDF Model uses concepts known from graph theory to represent subjects and the relations between them: any concept is represented by a *node*, and relationships are *arcs* between nodes. Since only *binary and directed* relationships are allowed in RDF, a single arc always connects exactly two nodes, the start node and the end node. These two nodes are known as *subject* and *object*, respectively, the arc is called *predicate* (or *property*). Together, the two nodes and the arc represent a single statement. A simple RDF statement is shown in Figure 3.5.
Since statements always consist of a subject, predicate and object, they are also referred to as *triples*. Instead of using the graphical display format or the RDF/XML notation, RDF statements are often displayed as triples using the *triple notation* as shown below:

```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740>␣.

<http://www.example.org/index.html>
<http://www.example.org/terms/creation-date>
```

```
"August␣16,␣1999"␣.

<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/language>
"en"␣.
```

Subjects have to be "resources" (i.e. concepts identified by URIs), whereas objects may either be resources as well, or literal values (see below). Predicates are again always identified by URIs that indicate the *type of relationship* expressed by the predicate (the example above uses relationships defined by the Dublin Core Metadata Initiative [WKLW98]).
While in principle subjects and objects are identified by unique URIs, it is sometimes convenient to establish concepts "on the fly" without having to create factitious URIs. RDF offers a construct called *blank node* which allows for creating anonymous concepts with no generally valid URI assigned. Instead, blank nodes are identified by arbitrary names that only have to be unique within the RDF document. With respect to the uniqueness of URIs, two resources with the same URI are generally considered to represent the same concept (see Section 2.2.2).

### Literals

In addition to connecting resources, RDF also offers the possibility to link concepts to simple data values, such as arbitrary strings, numbers, dates etc. These values are called *literal values*, or simply *literals*.
There are two ways to specify literals: either a literal is a *plain literal* containing only some text with an optional language tag, or it is a *typed literal* containing a value whose lexical form is determined by the mandatory datatype URI. The datatype URI corresponds to the datatypes defined in the XML Schema standard (e.g., `"27"^^<http://www.w3.org/2001/XMLSchema#integer>`).

### Containers

RDF facilitates the handling of containers and collections by defining a *container vocabulary* that consist of three different types of containers: `rdf:Bag`, `rdf:Seq` and `rdf:Alt`. They can be used as types for concepts that contain other things:

- a "bag" is a container for resources or literals with no inherent order of the members, possibly containing duplicates;
- a "sequence" is a container for resources or literals where the order of the members is important, possibly containing duplicates;

- an "alternative" is a container for resources or literals that are alternatives for a property value, again possibly containing duplicates.

However, the use of the predefined container vocabulary is not mandatory:

> "Users are free to choose their own ways to describe groups of resources, rather than using the RDF container vocabulary. These RDF containers are merely provided as common definitions that, if generally used, could help make data involving groups of resources more interoperable." [RPR04]

### Reification

RDF provides support for reification, i.e. making assertions about assertions. In contrast to Topic Maps, relationships between concepts are not represented through any "symbols" in RDF by default. Therefore, reification of RDF relationships is more complex than reification of Topic Map associations.
RDF provides four predefined properties, `rdf:Statement`, `rdf:subject`, `rdf:object` and `rdf:predicate`, that can be used to denote a RDF statement and its respective parts. If a RDF statement has to be reified, a new concept *representing that statement* is inserted into the graph. This entails the following transformations: the original triple (e.g. `ex:vienna ex:inhabitants "2000000"^^xsd:integer`) *is replaced* with 4 new triples: 1 triple for expressing the fact that the new concept is a statement (by using the predicate `rdf:type`), and 3 triples linking the original parts of the assertion to the predefined properties `rdf:subject`, `rdf:object` and `rdf:predicate`:

```
ex:newconcept rdf:type rdf:Statement
ex:newconcept rdf:subject ex:vienna
ex:newconcept rdf:predicate ex:inhabitants
ex:newconcept rdf:object "2000000"^^xsd:integer
```

Figure 3.6 shows a graph-based version of this example. As demonstrated, RDF reification is not as straightforward as reification in Topic Maps.
The most important drawback however is the need for *replacing* the original triple with its reified version. Generally, providing additional information about another statement is not expected to affect the original statement in any way; but this is not true for RDF reification. Therefore, any application that allows for reification not only adds information to the ontology, but actually *changes* existing statements.
Also, the reified version of a statement is much more abstract and has to be interpreted in a different way than ordinary statements, for instance by ontology query languages. This adds significant complexity to RDF applications and potentially limits the broad propagation and easy application of RDF reification.
Finally, opposed to Topic Map reification, RDF uses no *preemptive reification* by default. Instead, relationships are only reified *in situ* if and when it is needed. This so

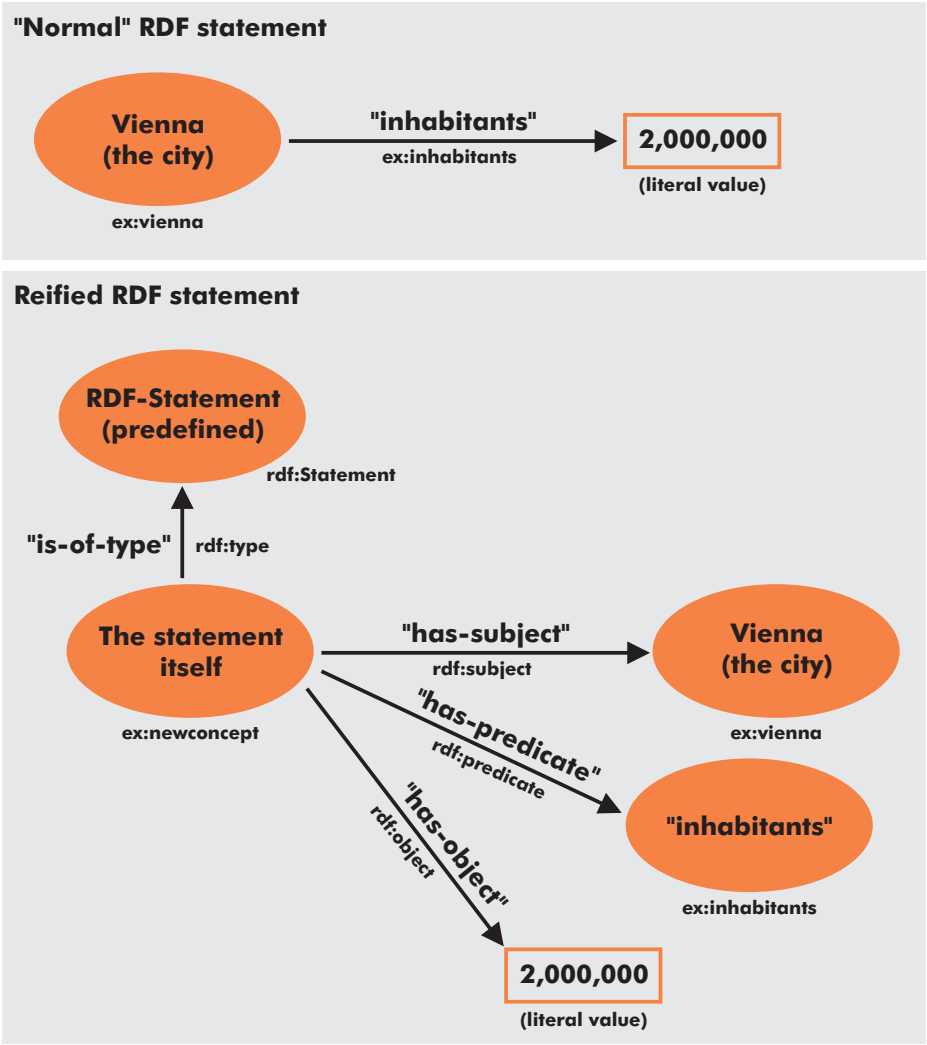


Figure 3.6: The reification process for RDF graphs.

called *lazy reification* has several consequences which can be considered problematic under certain circumstances.

First, upon reification of a relationship, it has to be decided whether the original relationship should be effectively *replaced* (and therefore removed from the RDF graph), or whether it should stay in place *in addition to its reified version*. The replacement of the relationship may e.g. not be suitable for "remote" ontologies, i.e. ontologies that are not under control of the reifying party. Also, reifying a large number of relationships by replacing them may entail performance issues and render the RDF graph overly complex.

On the other hand, leaving relationships in place is not an elegant solution either: changes to the relationship must be propagated to both the original *and* the reified version, the RDF graph becomes ambiguous and lacks parsimony. After all, lazy reification as used by RDF can be problematic with respect to its usage within Semantic Web ontologies.

### 3.2.2 RDF Schema

Regarding common ontological tasks such as instancing, subclassing and constraining of concepts, RDF itself has little to offer. Through the predefined predicate `rdf:type`, RDF allows for creating instances of classes, but even marking concepts as classes or creating subclasses is beyond the scope of RDF.

In order to define new RDF vocabularies (consisting of classes, class hierarchies and properties), the *RDF Vocabulary Description Language*, or *RDF Schema (RDFS)*, can be used [VDL04]. RDFS does not define any domain-specific classes or properties, but provides additional predefined RDF constructs that can be used to define such classes and properties. Thus, the most important RDFS additions to the RDF vocabulary are `rdfs:Class` and `rdfs:subClassOf` to create nodes which represent classes, as well as `rdfs:range` and `rdfs:domain` to constrain properties (the predefined property `rdf:property` is already defined in the RDF standard itself). `rfs:range` constrains the types or values that a property may accept, while `rdfs:domain` limits the usage of a property by defining the classes which a property may be used for.

RDFS also facilitates the creation of subproperties through the relationship type `rdfs:subPropertyOf`. Additionally, there is a number of other RDFS built-in properties like `rdfs:comment`, `rdfs:label` and `rdfs:seeAlso` play a minor role with respect to the ontological capabilities of RDF(S).

Finally, it should be mentioned that almost every RDF application uses both RDF *and* RDFS (or its extension OWL); in fact, the distinction between RDF and RDFS, while definitively existent in theory, is almost neglectible in practice.

### 3.2.3 The Web Ontology Language (OWL)

The Web Ontology Language (OWL) is a W3C standard which is built upon RDF Schema and enriched with semantics taken from DAML+OIL. Extensive descriptions of OWL are to be found for instance at the W3C site [OFT04,OGD04,OSM04,OUC04]. The history of OWL begins with the W3C standard RDFS which is described as "semantic extension of RDF" ( [VDL04]). RDFS is part of six documents published by the W3C that together constitute the RDF standard family. The vocabulary defined by the lightweight RDFS is however very limited and was extended by a semantic markup language named DAML+OIL[8]. DAML+OIL [CvHH+01] was first defined in 2000 and is a combination and advancement of the DAML-ONT ontology language (DARPA Agent Markup Language[9]) and the Ontology Inference Layer (OIL[10]). DAML+OIL offers a vocabulary for defining ontologies with constraints and inference mechanisms. OWL is basically an extension of DAML+OIL (for a list of changes from DAML+OIL, see [BvHH+04], Appendix D). There are three increasingly-expressive subsets of OWL, OWL Lite, OWL DL and OWL Full. OWL Lite "supports those users primarily needing a classification hierarchy and simple constraints", OWL DL[11] is for users "who want the maximum expressiveness while retaining computational completeness [...] and decidability" whereas OWL Full offers the "maximum expressiveness and the syntactic freedom of RDF with no computational guarantees" [OGD04].

#### Classes

All concepts with OWL are implicitly instances of the uppermost class `owl:Thing`. In order to allow for the creation of user-defined classes, OWL introduces the `owl:Class` property which is an extension of `rdfs:Class` and is used to denote classes throughout OWL ontologies. Since `owl:Class` is a subclass of `owl:Thing`, every user-defined class is also a subclass of `owl:Thing`, because the `rdfs:subClassOf` relationship is transitive. "Individuals" (i.e. concepts that are not classes) are thus direct members of `owl:Thing`, except if they are instances of other classes, obviously.
Since only OWL Full is capable of handling a concept as a class and an instance at the same time, [OGD04] recommends to clarify the intended usage of an ontology beforehand. If OWL Lite or DL are to be used (e.g. because of computational advantages), workarounds for such concepts must be arranged.
OWL is also able to represent *unnamed classes* (or *anonymous classes*) that are not

[8] See [OO02] for a good introduction to DAML+OIL.
[9] `http://www.daml.org/`
[10] `http://www.ontoknowledge.org/oil/`
[11] "OWL DL is so named due to its correspondence with description logics, a field of research that has studied a particular decidable fragment of first order logic. OWL DL was designed to support the existing Description Logic business segment and has desirable computational properties for reasoning systems." [OGD04]

represented explicitly by means of an URI, but instead implicitly by one of the following OWL constructs:

- an exhaustive enumeration of individuals that together form the instances of a class (`owl:oneOf`)
- a property restriction (there are 6 OWL constraints to specify value and cardinality constraints; see example given below)
- the intersection of two or more class descriptions (`owl:intesectionOf`)
- the union of two or more class descriptions (`owl:unionOf`)
- the complement of a class description (`owl:complementOf`)

The following example is provided by the OWL Guide [OGD04], with the element `<owl:Restriction>` being the unnamed class:

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
       1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Finally, there are two more built-in properties for further specifying classes:

- `owl:equivalentClass` to express the fact that two classes have exactly the same set of instances, although the classes themselves are not equal;
- `owl:disjointWith` to express the fact that two classes must not have any common instances.

#### Instances

There are three built-in OWL properties that allow for stating facts about the identity of individuals:

- `owl:sameAs` indicates that a certain concept is equal to some other concept; for OWL Full, the concepts invoved may be instances or classes; for OWL Lite and OWL DL, only instances are allowed.

- `owl:differentFrom` indicates that two instances refer to different subjects.
- `owl:AllDifferent` states that all instances in a list refer to different subjects; this construct can be replaced through corresponding binary `owl:differentFrom` properties and is hence provided for convenience only.

### Properties

In addition to the RDFS constructs related to properties, `rdfs:subPropertyOf`, `rdfs:range` and `rdfs:domain`, OWL defines two types of properties: `owl:ObjectProperty` and `owl:DatatypeProperty`. They (and their potential subproperties) are used to relate concepts to concepts, or concepts to datatypes (i.e. simple literals or typed literals), respectively. It should be noted that these types of properties are disjoint classes: either a property is subproperty of `owl:ObjectProperty`, *or* it is subproperty of `owl:DatatypeProperty`.
The most important additions of OWL with respect to properties however are the following *property characteristics*:

**Transitivity:** if a property P is an instance of type `owl:TransitiveProperty`, then for any x, y and z: P(x, y) and P(y, z) implies P(x, z).

**Symmetry:** if a property P is an instance of type `owl:SymmetricProperty`, then for any x and y: P(x, y) implies P(y, x).

**Functional:** if a property P is an instance of type `owl:FunctionalProperty`, then for any x, y and z: P(x, y) and P(x, z) implies x = z.

**Inverse:** if a property P is an instance of type `owl:inverseOf`, then for all x and y: P(x, y) implies P(y, x), and P(y, x) implies P(x, y).

**Inverse functional:** if a property P is an instance of type `owl:InverseFunctionalProperty`, then for all x, y and z: P(y, x) and P(z, x) implies y = z.

These property characteristics significantly improve support for inference engines and together with class hierarchy relationships form the backbone of OWL ontologies.

### Versioning

OWL provides several built-in properties to enable versioning of ontologies and parts theirof. `owl:versionInfo` allows to attach an arbitrary string to a concept which contains information about the current version of that concept. This feature is intended to be used by versioning systems. `owl:priorVersion` can be applied to an entire ontology and points to another ontology which represents a prior version.

`owl:backwardCompatibleWith` and `owl:incompatibleWith` may also be applied to entire ontologies and indicate the level of backward compatibility with other ontologies (which are assumed to be an prior versions). Finally, `owl:DeprecatedClass` and `owl:DeprecatedProperty` designate deprecated classes and properties; concepts marked this way should not be referred to any more. This allows for slowly migrating applications to new versions of ontologies.
Finally, OWL also enables the import of other ontologies "whose meaning is considered to be part of the meaning of the importing ontology" [BvHH+04]. The import is achieved through use of the transitive `owl:imports` property.

## 3.3 Comparing Topic Maps and RDF

This section compares the constructs provided by Topic Maps and RDF; an exhaustive analysis of the two technologies on a data model level can be found in Lars Marius Garshol's paper "Living with Topic Maps and RDF" [Gar03a]; Steve Pepper, on the other hand, gives some general indications on the relationship between Topic Maps and RDF [Pep02b]. Some hands-on experiences with using both Topic Maps and RDF to create an index of conference papers are described in "Lessons on Applying Topic Maps" [Pep02a].

### 3.3.1 Concepts

Topic Maps and RDF are both "identity based technologies", which means that they both primarily deal with representing subjects and making assertions about subjects. Terminology is somewhat confusing here: what Topic Maps call "subjects" is exactly the same as RDF's "resources"; RDF "resources" are therefore *not* limited to electronic resources, but may represent anything whatsoever.

### 3.3.2 Assertions

Topic maps offer three methods to make assertions about subjects: names, occurrences and associations. RDF knows only the concept of "statements":

- *Names* are added to RDF resources by using a name property (which must be defined by the vocabulary used) instead of being assigned to subjects as "privileged" (and built-in) properties.
- *Occurrences* are represented almost identically in Topic Maps and RDF ("properties").
- *Associations* on the other hand do not have a direct counterpart in RDF. Topic Maps associations can only be used to connect topics (but not topics and resources), and may have any number of roles (and respective role players). RDF

only allows exactly 2 roles (the subject and the object of a statement). Finally, associations do not imply any directions, whereas RDF statements always point from the subject to the object of an assertion (this is also the reason for the existence of constructs like `owl:inverseOf` in OWL).

### 3.3.3 Reification

As shown in Section 3.1.7, reification of assertions can be achieved easily and elegantly with Topic Maps. RDF also supports reification (Section 3.2.1), but the author has to take a rather awkward way: the original statement has to be replaced by an empty RDF node which is given the type `rdf:Statement` and which uses a special vocabulary to add its subject, object and property; reified statements must therefore be treated differently as ordinary RDF statements.

### 3.3.4 Qualification

Topic maps have the built-in feature of *scopes* that allow qualification of topic characteristics, i.e. to express the context or limited validity of an assertion (see Section 3.1.6). RDF has no comparable feature, except a language identifier that may be attached to literals. Of course, qualifying statements can be made if statements are reified first, but for RDF statements reification is rather unhandy, as noted above. Lars Marius Garshol identifies the underlying problem as follows:

> "The key problem here is that statements in RDF have no identity, which means that it is impossible to make resources that represent them (without changing the statements) and since the model does not directly support qualification support for qualification cannot be added through reification. This is one of the most fundamental differences between Topic Maps and RDF, and one that has so far frustrated all attempts to model Topic Maps in RDF in a natural way." [Gar03a]

### 3.3.5 Types and Subtypes

Both Topic Maps and RDF provide similar ways to create class (or type) hierarchies and class instances (see Section 2.4). For RDF, creating subclasses is supported only by using RDF Schema (RDFS), or OWL.

## 3.4 Summary

With respect to Topic Maps and similar technologies like RDF, an ontology can be used to handle the following core tasks:

1. It is a *container for classes* and provides methods to establish relationships between these classes ("class hierarchy", see Section 2.4). As this is the most obvious purpose of an ontology, many definitions of the term "ontology" only cover this aspect.

2. In most cases, ontologies also contain instances of classes; these instances can then be connected through relationships ("associations" or "statements").

3. It can be used to *express constraints* for the classes within the hierarchy; this combination of class hierarchy and constraints, makes is possible to validate classes and instances against it and to ensure thereby the *semantic* correctness of the ontology.

4. It offers definitions of certain "built-in" properties for classes and relationship types that are necessary to allow *reasoning* (or inferencing) on the class hierarchy and the instances in the ontology. This makes it possible to get information from an ontology which has never been supplied directly: e.g. if an ontology contains the information that the concepts "mother" and "father" are subclasses of the concept "parent", and that "Leopold Mozart" was the father and "Anna Maria Pertl" the mother of "Wolfgang Amadeus Mozart", a query against that ontology for Wolfgang Amadeus Mozart's parents should retrieve both his father and his mother, although the fact that Leopold Mozart is Wolfang Amadeus' *parent* has never been stated directly.

Smaller ontologies for demonstration purposes can be found for instance at the ontology library sites [DOL04, OLB], including ontologies for wines, travel and tourism, countries etc.

# 4 Implementation Challenges

This chapter deals primarily with issues regarding the practical implementation of applications that use ontologies as knowledge repositories. Three key topics were found to be especially challenging: providing scalable, persistent storage, allowing for simple, yet flexible and powerful querying, and deducing new information by automated reasoning over knowledge bases. Interestingly, the former two of these challenges are generally considered to be solved for traditional applications, but appear anew if ontologies are involved. Also, solutions for Topic Maps and RDF/OWL sometimes follow quite different approaches and show individual strengths and weaknesses that will also be described in the following sections.

## 4.1 Persistent Storage for Ontologies

### 4.1.1 Overview

Ontologies represented with RDF/OWL and Topic Maps are likely to become valuable tools for knowledge management and are necessary for the *semantic interchange* of data between autonomous agents in a distributed environment such as the proposed Semantic Web. RDF, OWL and Topic Maps are relatively new standards, which gives them the possibility to take advantage of the experiences made with large, distributed information stores such as the WWW. However, by the time of writing only few large-scale implementations of RDF/OWL or Topic Maps exist outside of tightly controlled research areas.

While ontologies are certainly capable of representing knowledge, this knowledge must also be made accessible to users and agents. Therefore, persistent storage and powerful query mechanisms for ontologies are critical issues for any ontology framework. Providing scalable storage facilities is even more important as most example ontologies available for the new ontology representation formats are quite simple and can easily be kept in memory by ontology management systems. Real world ontologies however are expected to grow up to significant size, depth and complexity, even more if they are all connected through a future Semantic Web.

Compared to traditional web pages which form a large part of the world's most successful information store, the WWW, ontologies are unequally more difficult to represent, store and query. The simple HTML format of web pages and their organization as basic textfiles in a filesystem is certainly inadequate for a large and complex ontology.

Ontologies in textual format (e.g. RDF/XML or XTM) are indeed available for download on dedicated web sites, but if ontologies were actually to be distributed this way, large ontologies would for instance have to be partitioned into many small files, like the "RDF dump" of the DMOZ Open Directory [DMO]. This is however not an option if one considers the large number of interconnections between concepts in complex ontologies; the DMOZ dump for instance contains about 300MB of data in compressed format.
Rather, with respect to storage and data exchange, ontologies should be regarded as *repositories* [Ahm00] that allow for adding, deleting, updating, and retrieving relevant information through appropriate APIs or languages for data manipulation and querying. This perception permeates the existing efforts for establishing persistent storage facilities.
The perception of ontologies as repositories also makes them comparable to traditional relational databases with respect to the features provided by storage services and the expressiveness of query languages. Different approaches for enabling persistent and scalable storage for ontologies are described in the next section; query languages and inference mechanisms are discussed in sections 4.2 and 4.3.

### 4.1.2 Approaches to Providing Persistence

Both OWL and Topic Map ontologies use multiple notations such as RDF/XML or XTM for information exchange purposes. However, as already mentioned earlier, these formats are merely serialization formats of the underlying, abstract data models that consist of nodes and arcs in case of RDF or topics, associations, occurrences and scopes etc. (in case of Topic Maps). In order to provide persistent storage for these ontology representation formats, basically three different approaches exist:

1. The *concepts represented in the ontology* are considered as individual, different entities and are thus mapped to according structures, usually in a traditional relational database system. The layout of these structures complies with the properties of the respective concepts.

2. The *constructs of the underlying data model* are regarded as individual entities which are to be persisted, usually again as structures in a relational database system. The layout of the structures corresponds to the properties of the entities of the data model as they are defined by the respective standards (W3C RDF/OWL or ISO 13250 Topic Maps). For RDF, the simple three-valued statement data model is often used as basis for dedicated so called "triple stores".

3. The *serialized versions* of the concepts in the ontology are used for persistence purposes. As both Topic Maps and RDF offer XML based notations, ontologies can be represented as pure XML documents, which in addition are expected to correspond to the according DTDs defined by the respective standards.

### Data Centric Persistence

The first approach is also known as *data centric* approach and often mentioned in context with mapping XML documents to relational databases[1] [KK03, Bou01a, Bou01b, Mit03]. With respect to ontologies, the process can be described as follows.
The first step is to identify the types of concepts and their properties that are to be stored in the ontology. Then, these types of concepts are mapped to according tables in a traditional RDBMS, with the previously identified properties being the fields of the tables. Finally, the instances of the classes can be inserted into the tables as rows, with one row representing one instance of a concept. This procedure is the same for subjects, relationships and all other data model entities defined by the respective standard.
In addition, several "auxiliary" tables are needed to keep track of whether a certain table maps to a subject or to a relationship etc. This leads to the situation that the database is actually split into two "virtual layers": the virtual "schema layer" consists of the auxiliary tables that keep track of all classes in the ontology, whereas the virtual "data layer" contains the tables created as instance containers for specific classes.
Such a data centric approach was for instance originally followed by the Sesame ontology framework [BKvH01, BKvH02] in conjunction with a PostgreSQL database. Figure 4.1 shows the setup of the Sesame data centric object-relational mapping for an exemplary ontology covering books and their writers (represented by the *Book*, *Writer* and *FamousWriter* classes) which are connected by *hasWritten*-relationships.
There are two advantages that can be exploited with the data centric approach. First, query answering as well as inserting, removing and updating instance of classes is extremely inexpensive and straightforward, as there is virtually no difference to traditionally designed databases. All manipulations concerning instances are in effect nothing more than executions of data manipulation commands as they are natively provided by all RDBMS. Second, some RDBMS like PostgreSQL offer built-in object-relational features that can be used directly for modelling class-subclass relationships etc. PostgreSQL databases offer for instance the possibility to create subtables that are connected to their parent tables through transitive relationships. This allows for creating a table for a certain class and according subtables (for subclasses of that class). The same is true for properties and subproperties, accordingly.
The main drawback of the data centric approach is that changes to the class hierarchy in an ontology are extremely expensive, as they require creating new entities in the database. For every new class (and also subclass) that is to be inserted into the ontology, a respective table has to be created, even if only a small number of instances is present. This means, that changes to the class hierarchy always require data definition commands to be performed, which are expensive in almost any RDBMS. Referring to the discussion in Section 2.5, in most cases an ontology (or the class hierarchy contained

[1] In this context however, XML and mapping methods are not directly addressed.

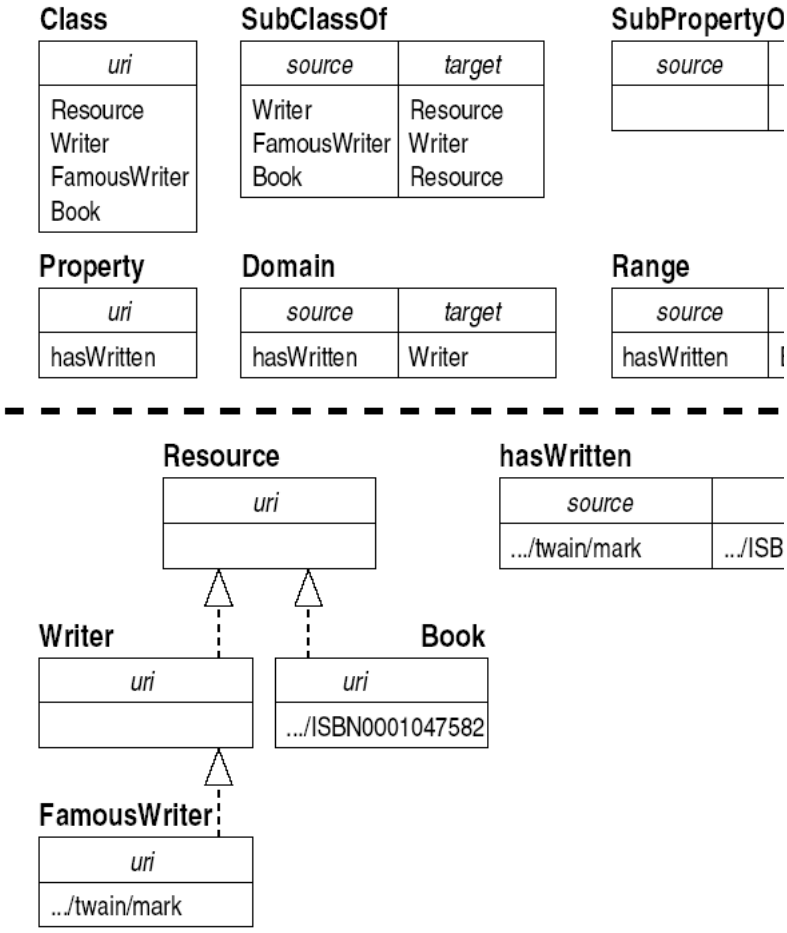


Figure 4.1: The Sesame data centric object-relational mapping [BKvH01].

in it) can not be considered to be static; rather, one of the main strengths of ontologies is to facilitate the use of highly flexible schemas (if a variable class hierarchy can be considered to be a schema at all).
Therefore, with exception of a few, special cases, changes to class hierarchies in ontologies must be assumed to happen frequently, which makes the data centric approach unfeasible for many applications. Another reason for abandoning the data centric object-relational approach in real world applications is its inflexibility against *incremental uploads* of large datasets, as the developers of Sesame noticed:

> "Our experiences with this database schema on PostgreSQL were not completely satisfactory. Data insertion is not as fast as we would like. Especially incremental uploads of schema data can be very slow, since table creation is very expensive in PostgreSQL. Even worse, when adding a new subClassOf relation between two existing classes, the complete class hierarchy starting from the subclass needs to broken down and rebuilt again because subtable relations can not be added to an existing table; the subtable relations have to be specified when a table is created. Once created, the subtable relations are fixed." [BKvH02]

Summarized it can be said that the data centric object-relational approach of providing persistence for ontologies, while certainly being an interesting and promising one, in practice fails to deal with variable class hierarchies in an efficient way, which is usually reagarded as a crucial feature of ontologies.

### Structure Centric Persistence

The second approach is also known as *structure centric* and is equally popular among Topic Map and RDF implementations. As it is the case with the data centric approach, persistency is eventually provided by a traditional RDBMS, but usually without requiring object-relational features. Opposed to the first approach, the key idea here is to map the finite number of *data model concepts* to according structures (tables) in the relational database. Again, the process has also been described for XML documents [KK03, Bou01b, Mit03, Bou01a] but has also been specifically implemented for both Topic Map and RDF applications.
As outlined in Section 3.1, the Topic Map data model offers a small number of built-in concepts, like Topic, Association, Occurrence, Scope etc., whose properties are well defined. In contrast to the actual classes and instances they *represent*, the number and design of these built-in concepts are static (as they are standardized). Therefore, it is a straightforward task to create corresponding structures in a RDBMS and map the concepts to these structures in such a way that in the end there is one table for all topics, one table for all associations, etc. Various examples of this implementation for Topic Maps exist, e.g. [WM01, KLS+01].

| subject | predicate | object |
|---|---|---|
| ex:vienna | ex:inhabitants | 2000000 |
| ex:mozart | ex:born-in | ex:salzburg |
| | | |
| | | |
| | | |
| | | |
| | | |

Figure 4.2: A naively implemented triple store table.

With respect to RDF, the data model consists basically only of statements, with each statement including a subject, an object and a predicate (Section 3.2). This means that for a naive approach only one single table (with three corresponding text fields containing the respective URIs or literals) is needed to express a complete RDF graph, as shown in Figure 4.2. Due to the layout of their tables, databases configured this way are therefore commonly referred to as *triple stores*. They are certainly a very elegant solution for ontology persistence and are probably one of the main reasons that RDF/OWL has gained significant popularity among ontology developers. Also, many variations and improvements over the naive approach exist, mainly in order to achieve high levels of scalability.

The first advantage of the structure centric approach is its ability to allow for inexpensive, frequent changes of instance data as well as of schema information (class hierarchies). Since all assertions, including hierarchical relations, are broken down to the level of single statements, no artificial distinction between "schema layer" and "data layer" has to be made. This allows not only for representing frequently changing ontology hierarchies, but also for efficient incremental incorporation of large datasets, as no structural changes of the underlying database schema are required.

The second advantage of structure centric ontology representation is commonly found to be reported for dedicated triple stores, but also applies for Topic Map representations. Due to the fixed, rather simple architecture of the database, scalability optimizations are easy to apply, enabling the efficient storage of millions of concepts and relationships. Several optimizations that were realized in practice are outlined in Section 4.1.3.

One main disadvantage of the structure centric approach (in the case of RDF triple stores) is encountered when retrieving statements for answering ontology queries. In order to evaluate a condition that not directly addresses the URIs or literals of the statements to be retrieved, the table containing the statement triples has to perform one or more *self-joins*, an operation which is expensive for large datasets [ACKP01, KK03]. Such large datasets must be considered to occurr frequently, as *all information*

of an ontology is stored within a single triple table. It is therefore not uncommon for such a table to contain millions of triples, which have to be compared to each other even several times, depending of the nature of the query to be answered. Although various optimization efforts try to limit the negative effects of storing triples in a single table, generally worse query answering performance has to be expected compared to the object-relational approach.
One more positive potential of structure centric persistence should be mentioned. First, in addition to querying and modifying the contents of an ontology through traditional SQL statements like SELECT, INSERT, UPDATE or DELETE, many modern RDBMS also provide support for *storing and retrieving XML-formatted data*. Usually, XML documents are required to conform to a predefined schema (e.g. a DTD) and are then mapped to according relational structures within a database. This functionality can only be exploited as long as the schema of the XML documents is not changing (frequently), because the mappings are usually regarded as static. The structure centric approach of storing ontologies meets this criterion, thus enhancing the interoperability between ontology applications and databases by facilitating the use of XML-based data exchange.
Similar considerations can be applied to frameworks that provide persistence for Java objects etc., such as Hibernate [Hib], Castor [Cas] and many others. Again, rather static mappings are being used, which asks for a static database model. Some frameworks actually provide some automated creation of database schemas, but this is again not an option for frequently occurring schema changes. Therefore, the structure-centric approach is also the only one appropriate for such frameworks, apart from the fact that most object-oriented applications which capsulate ontologies (e.g. Topic Map engines etc.) will generally use the standardized RDF or Topic Map data models as design patterns for their internal classes (whose instances eventually are the objects to be persisted).

#### XML-based Persistence

The third approach to achieve ontology persistence is inspired by the XML serialization formats of Topic Maps and RDF, namely XTM and RDF/XML. For instance, the XTM4XMLDB [Lis] implementation provides persistence for Topic Maps using the open-source XML database eXist [eXi].
From the perspective of XML processing tools and applications, both XTM and RDF/XML are just ordinary XML schemas (or DTDs) which impose strict constraints on the structure of XML documents containing serialized ontologies. Therefore, it seems to be a straightforward task and promising idea to store XTM and RDF/XML documents within dedicated XML databases, especially from the perspective of an application developer. After all, quite an amount of work is needed to provide efficient persistence for ontologies using relational databases as outlined in the two former

approaches.
As Ronald Bourret points out, "the term 'native XML database' first gained prominence in the marketing campaign for Tamino, a native XML database from Software AG" [Bou03], but no formal technical definition has ever been provided. Opposed to XML-enabled relational databases, native XML databases completely abandon the concept of tabular organization of data in favor of text-based or model-based storage mechanisms specifically tailored to XML:

> "*Native XML databases* are databases designed especially to store XML documents. Like other databases, they support features like transactions, security, multi-user access, programmatic APIs, query languages, and so on. The only difference from other databases is that their internal model is based on XML and not something else, such as the relational model." [Bou03]

With respect to their use as repositories for ontologies, one important observation must be taken into consideration: XML databases primarily provide methods for storing and retrieving *XML documents*. Although access to arbitrary parts of documents is also possible, efficient retrieval of data is mostly achieved when accessing either whole documents or their parts *in the order of their appearance* (within the document hierarchy): "retrieving data in the hierarchy in which it is stored is very quick, but retrieving the same data in a different hierarchy is not" [Bou03]. Also, many native XML databases, while supporting large numbers of documents, require individual documents to be rather small in size for efficient processing.
The focus on documents as primary entities poses the question of how XML documents relate to ontologies. Although one single XML document can contain a complete ontology from a conceptual point of view, storing such a large, single document is not advisable for the reasons given above. Rather, ontologies have to be broken down into multiple XML documents, for instance based on the concepts and relationships that are represented, or on the entities of the underlying data model (RDF nodes and arcs, Topic Map topics, associations etc.). However, there are no standardized guidelines for such partitionings of ontologies.
In other respects, native XML databases provide features that are not necessarily needed by ontology applications. The most important one is that XML databases are able to store XML documents *in the exact format they were specified*, including comments, blank lines, processing instructions etc. This makes it possible to *round-trip* documents, which means that storing and retrieving a document results in exactly the same data. This ability is crucial for many XML processing applications, but is not generally required for storing ontology serializations.
In fact, XML documents are mainly used for two different purposes: either they serve as pure containers for data and are thus said to be *data-centric*[2], or they encapsu-

[2]The term "data-centric XML document" as used in this place is not related to the "data-centric

late text intended for human consumption with additional markup and are called *document-centric* (because they are comparable to text documents generated by traditional word processors).

Document-centric XML documents "are characterized by less regular or irregular structure, larger grained data [...], and lots of mixed content" [Bou03]. The order of the elements within the document hierarchy is usually significant. Data-centric documents, on the other hand, exhibit a very regular structure with fine grained data elements without mixed content. Since data-centric XML documents are intended for machine consumption, the order of the elements in the document hierarchy is not important in most cases.

Obviously, XTM and RDF/XML files are data-centric documents, as they are merely serialized versions of ontologies which are not conceptually tied to XML. Native XML databases however must support both data-centric and document-centric XML files, which results in a certain overkill of features that are not needed for storing XTM or RDF/XML documents.

Finally, native XML databases are not superior to relational databases in terms of support for the ontological data models used by Topic Maps or RDF. While these data models are both based on *graphs*, XML uses *tree-based* structures, which are great for applications where tree structures are natively used, but provide almost no advantages for representing graphs. This can also be recognized by the fact that XML-based query languages, such as XPath or XQuery, offer no additional power over SQL for almost any operations on ontologies, whereas they are definitely superior whenever "genuine" tree structures are to be queried: "questions like 'Get me all documents in which the third paragraph after the start of the section contains a bold word' [...] are clearly difficult to ask in a language like SQL" [Bou03]. Relational databases use a model consisting of tables and rows, which neither allows for representing graphs directly, but has proven to be reliable and, above all, massively scalable.

Summarized, native XML databases are certainly interesting candidates for storing ontologies serialized with XML-based notations. However, with respect to relational databases, they offer no real advantages on the representation of the underlying ontology data models, while at the same time features like round-tripping and hierarchical queries (e.g. using XQuery) are not needed by data-centric XML documents such as XTM or RDF/XML files. Also, frameworks that provide persistence for ontologies will probably prefer relational databases over native XML databases because the former are not only more "mature", but there are also many different RDBMS implementations ranging from simple in-memory databases to enterprise server RDBMS, covering all major platforms and operating systems and thus facilitating portability and customization.

approach" for ontology persistence described earlier.

### 4.1.3 Scalability Considerations

If ontologies are to become the backend of semantically enriched applications, they will surely need to represent many thousands or even millions of concepts and relationships. Therefore, the scalability of ontology data stores is an important issue for any larger ontology-powered application.

As Section 4.1.2 demonstated, there are not many options that promise to scale beyond a certain depth and complexity of an ontology. The only approach to provide persistence in a truly scalable way is the structure centric one, which for RDF results in dedicated triple stores. Topic Maps can be persisted by using a similar schema, but due to the higher complexity of the Topic Map data model with respect to RDF, more structures are needed in a database schema. While quite a large number of Topic Maps applications are known to be successfully working under real world conditions, the lack of information about Topic Maps and scalability should be noted here; certainly, this crucial aspect has to be examined more thoroughly. A theoretical work considering Topic Maps as hypergraph [BA01] has been presented in 2001, but no results of real world implementations are available by the time of writing. Topic Maps concepts have also been used to develop UTON, "a technology framework for building large scale ontologies" [Var02], but again results can not directly be interpreted for Topic Maps.

Many RDF triple stores, on the other hand, have been tested with large quantities of data to determine their ability to scale with large ontologies [Lee04]. Instead of presenting various implementations and their respective performance figures, some general approaches on improving scalability for dedicated triple stores are described here. As starting point, the naively implemented triple store table as shown in Figure 4.2 is used.

The first, quite obvious improvements are motivated by the long and verbous character-based format of URIs. URIs used in RDF/XML are ordinary XML URIs which consist of namespaces and identifiers for the resources that are represented. Since namespaces are usually used many times in different URIs, database size can be reduced by moving them to a separate table. Sometime, this idea is extended to whole URIs, as e.g. proposed by ICS-FORTH's "generic representation" [ACKP01]. The "generic representation" (shown in Figure 4.3) consists of two tables, one storing the actual URIs and assigning them unique (integer) numbers, the other representing the triples by using the unique numbers instead of full URIs. Triple stores configured in such a way are commonly referred to as *normalized triple stores*.

Another optimzation which is common among RDF triple stores is to treat literals different from URIs. This is usually achieved by using a dedicated column in the triples table, as it can also be seen in Figure 4.3. Since literals can only be used as objects (and never as subjects or predicates) in a RDF statement, only one of the two columns "object" and "literal" is used per row. Although this results in a large number of NULL values to be inserted, it increases scalability, because the distinction

**Triple table**

| predid:int | subid:int | objid:int | objvalue:text |
|---|---|---|---|
| 6 | 2 | 1 | |
| 5 | 3 | 7 | |
| 5 | 1 | 8 | |
| 5 | 9 | 2 | |
| 3 | 9 | | SunScale |

**Resource table**

| id:int | uri:text |
|---|---|
| 1 | http://www.dmoz.org/topics.rdfs#Hotel |
| 2 | http://www.dmoz.org/topics.rdfs#Hotel_Direc |
| 3 | http://www.oclc.org/dublincore.rdfs#title |
| 4 | http://www.dmoz.org/schema.rdf#Ext.Resource |
| 5 | http://www.w3.org/1999/02/22-rdf-syntax-ns#type |
| 6 | http://www.w3.org/2000/01/rdf-schema#subClassOf |
| 7 | http://www.w3.org/1999/02/22-rdf-syntax-ns#Property |

Figure 4.3: Outsourcing URIs for increased scalability [ACKP01].

between URIs and literals can be made already at database level and does not have to be performed by the application. There is also another reason for having a separate column for literals only: literals are simply strings that must be interpreted by the application (in case of typed literals, they are accompanied by a datatype identifier to facilitate interpretation) and can therefore also contain *URIs as values*. These URIs are not to be treated as "resources", but only as literal string values; however, they look exactly like resources identified by URIs, which makes literals and resource URIs indistinguishable from each other if they are not contained in separate table fields.

A novel approach to achieve persistence and scalability for RDF was developed by R. V. Guha and resulted in his triple store database *rdfDB* [Guh]. rdfDB uses *hashes of URIs* to build three B-Tree index files based on the URIs of "subject and predicate", "object and predicate", and "predicate" only, respectively. The idea of creating hashes from URIs has been adopted by other implementations, e.g. *3store* [RTS]. 3store uses the MySQL RDBMS for persistence and a database layout as shown in Figure 4.4. The "triples" table stores only the hashes of the URIs, whereas the tables "resources" and "literals" connect hashes and URIs (for resources) or hashes and literal values (for literals). The most appealing benefit from using hashes is that the lookup-tables "resources" and "literals" are only needed for converting *output-hashes* into URIs: the hashes of URIs within the query can be calculated by a hashing algorithm, and join operations in the database during query execution are performed with the hash values

**Triples**

| model | subject | predicate | object | literal | inferred |
|---|---|---|---|---|---|
| int64 | int64 | int64 | int64 | boolean | boolean |

**Model**

| hash | model |
|---|---|
| int64 | text |

**Resources**

| hash | uri |
|---|---|
| int64 | text |

**Literals**

| hash | literal |
|---|---|
| int64 | text |

Figure 4.4: The 3store database layout using hashed URIs.

themselves; therefore, only those hashes that appear in the result of a query must be re-transformed into URIs using the lookup-tables.
All implementations discussed so far treat all types of concepts and relationships equally, whether they are predefined in the respective standard or "user-defined", e.g. created through appropriate subclassing and instancing of basic elements. Although they are not different from a technical point of view, most ontologies will make heavy use of the built-in types and properties as defined by the Topic Maps or RDF/OWL standards. Therefore, efficient processing of these types and properties is especially important. Such considerations led to the development of so called *hybrid solutions*, for instance the "specific representation" of ontologies presented by ICS-FORTH (see Figure 4.5):

> "In the former representation [...] the core RDF/S model is represented by four tables [...], namely, `Class`, `Property`, `SubClass` and `SubProperty` which capture the class and property hierarchies defined in an RDF schema. [...] The main goal of SpecRepr is the separation of the RDF schema from data information, as well, as the distinction between unary and binary relations holding the instances of classes and properties." [ACKP01]

Another scalability improvement is introduced by the Jena Semantic Web Framework [Jen]. Jena2[3] uses a *denormalized triple store* for increased scalability, in contrast to the normalized triple store approach described earlier:

[3] Only the current version of Jena, Jena2, uses the denormalized version; its predecessor, Jena1, used a normalized triple store.

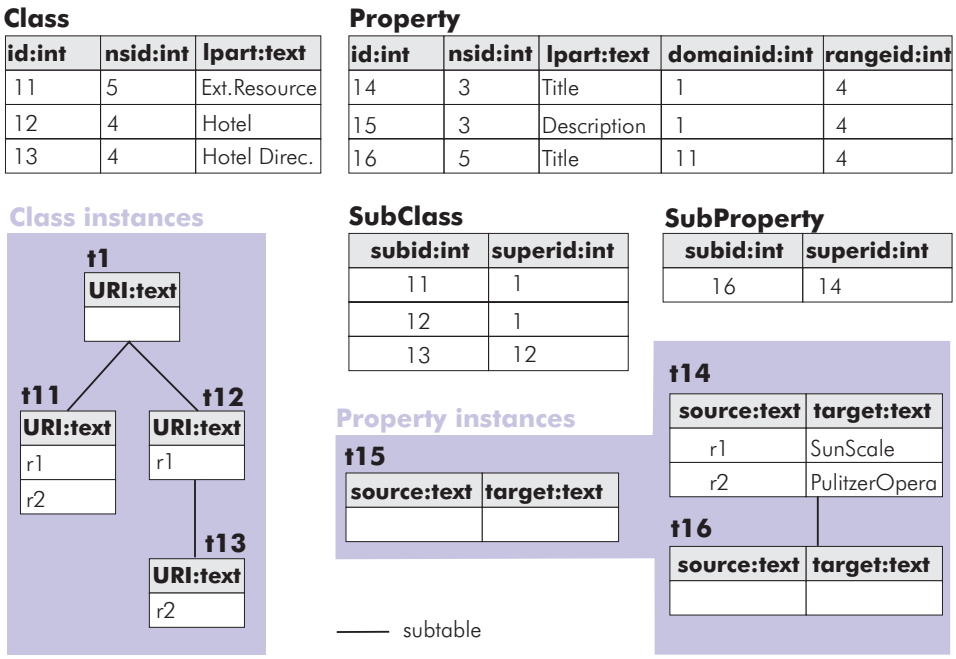


Figure 4.5: The hybrid solution, distinguishing RDF/S types and properties.

> "This normalized scheme uses less space than the standard triple store approach since a literal value or resource URI is only stored once, regardless of the number of times it occurs in statements. The space savings comes at a cost however since retrieving the subject, predicate and object values for a statement requires a 3-way join of the statement, literals and resources tables.
>
> [...]
>
> Jena2 stores RDF statements using a denormalized triple store approach which is a hybrid of the standard triple store and the normalized triple store. This scheme uses a statement table, a literals table and resources table as before. However, the statement table may contain either the values themselves or references to values in the literals and resources tables. 'Short' literals are stored directly in the statement table and 'long' literals are stored in the literals table. Similarly short URIs are stored directly in the statement table and long URIs are stored in the resources table." [Jen04]

The denormalized triple store approach taken by Jena2 is basically a trade-off between storage size and query response time: shorter response times are achieved at the

expense of additional storage space. A configurable threshold length separates "long" values from "short" ones, effectively limiting the negative effects on storage size and query response time.
Finally, most implementations providing persistence for ontologies take special care of "non-standard situations": the most commonly appearing exceptions are *reified statements* and *inferred assertions*. Reified statements are either flagged by an additional (boolean) column in the triple store, or are stored in a separate table (e.g. in Jena2): "the benefit of the reified statement table is that it stores reified statements in an optimized form" [Jen04], effectively removing the need for storing the four statements (that together form the reified version of a single "normal" statement) as four rows in a standard triple store (see also Section 3.2.1).
Inferred statements are usually automatically added by inference engines upon insertion of an asserted statement and must be marked as such, since they must be removed or recalculated whenever statements are deleted from the ontology. Without such indication, there would be no way to later on distinguish between statements that were explicitly inserted into the ontology and statements that were inferred automatically based on inference rules. Of course, marking inferred statements only applies to forward-chaining inference mechanisms which calculate inferred statements in advance (see Section 4.3).

### 4.1.4 Cyc Transcript Files

An interesting solution that is quite different from the approaches presented earlier is applied by the Cyc System [Cyc], which basically adopts an in-memory approach and adds support for updating the knowledge base by writing new and updated data to a *transcript file*:

> "The world file contains a copy of the knowledge in the KB that has been translated into a compact, efficiently loaded binary format called CFASL.
>
> [...]
>
> When the executable file is invoked to start a running process, part of the information that must be provided (e.g., via a command line argument or in a configuration file) is the name of the desired world file. The newly started process will try to locate the specified world file and, if successful, will try to load the entire contents of the file into memory (RAM plus virtual memory).
>
> [...]
>
> Every running Cyc image has a separate copy of the knowledge base. This raises the question of how, at a site where several people are using Cyc and the KBs of several images are frequently modified, the different KBs

> are kept in sync. This is accomplished by having each image write its own operations (KB modifications) to a sequence of transcript files. [...] a Cyc image that is completely up-to-date (i.e., that has processed all of the operations contained in the master transcript file) should be used to write out a new world file. This world then becomes the new foundational knowledge base that all Cyc images load when they start up [...]" [SGKM]

Transcript files as described by the Cyc architecture are appropriate if several users are modifying a large ontology: inconsistencies due to concurrent updates are detected whenever a Cyc image processes changes that were originally performed on another image, which is quite different from how RDBMS work (inconsistencies are prevented by locking mechanisms). Transcript files are therefore adequate for occasional updates with low probability of concurrency, but are likely to become too inflexible for other scenarios.

### 4.1.5 Persistent Storage Summary

As shown in the previous sections, traditional relational databases still play an important role for providing persistence for ontologies, even though they do not naturally support flexible schemas and hierarchical structures. By concentrating on the limited number of concepts defined by the Topic Maps and RDF/OWL standards, the effects of the variable nature of ontological schemas can be compensated, which enables sufficially scalable solutions for storing ontologies. Many optimizations and variations exist to further enhance speed and efficiency, most of them focusing on RDF triple stores.
The approaches presented in the previous section all share common features of modern RDBMS, such as multi-user access, ACID[4] transactions etc. Of course it is also possible to store ontologies in simple XML files, but this strategy is only advisable for small, single-user test environments and certainly does not scale well. The same is true for in-memory implementations, which are quite common among ontology frameworks. While offering superior speed when answering queries, multi-user access is usually provided for read-only scenarios only, if at all. Finally, the in-memory method employed by Cyc, while certainly being an interesting option, is not likely to perform well within distributed environments.

## 4.2 Querying Ontologies

As discussed in Section 1.3, representing knowledge with ontologies is an important step towards true computer-aided knowledge management. With the possibility of persisting large ontologies in appropriate storage facilities, ontologies would finally be

[4] Atomic, Consistent, Isolated, Durable

able to elvove from pure research objects to real world applications. However, representing knowledge is not enough to allow for practical use of ontologies: appropriate querying methods are equally important and must be provided to fully exploit the power of ontologies.
As there exist different approaches on storing ontologies, e.g. relying on RDBMS or native XML databases, one could consider the use of well-established query languages like SQL or XQuery [XQR04] as a logical consequence. Since ontologies can be decomposed into according tables in a relational database, or serialized to XML using one of the XML-based serialization formats, such query languages obviously offer the possibility to retrieve all data contained in an ontology. Such considerations are indeed not incorrect, but do not account for the special semantics used within ontologies.
Both SQL and XQuery are well suited for retrieving data from a *known schema* and are thus bound to the structure of the data store. A query author is generally expected to know this structure, a requirement that is acceptable for traditionally designed databases. Ontologies however actually use a data model that differs significantly from its representation in a database, as explained in Section 4.1. Users querying ontologies can be expected to know the abstract data model (and its primary elements, such as topics, associations, occurrences, or nodes and arcs) that Topic Maps or RDF use, but they should not have to deal with the implementation details of the ontology store (e.g. whether a data-centric or a structure-centric approach is followed, or which serialization syntax is used etc.). For XML documents, things are even more complicated, as many different XML syntaxes are allowed by the RDF/XML notation itself. Broekstra, Kampman and van Harmelen point out that

> "[...] the XML syntax for RDF is not unique: different ways of encoding the same information in XML are possible and in use currently." [BKvH01]

The ambiguity of the RDF/XML notation therefore effectively prohibits querying XML documents on a syntactical level. The solution of this problem is to distinguish between the *structural level* at which the information is encoded, and the *semantic level* which corresponds to the abstract data model used for ontology representation. As both SQL and XQuery clearly belong to the structural level, the need for dedicated query languages *at the semantic level* arises: "a direct usage of the XML syntax (and its underlying tree data model) in order to represent RDF/S (meta)data is not appropriate, since semantically equivalent descriptions may have several XML serializations" [KMA+02]. The purpose of such semantic query languages is to hide the implementation details from the ontology user and instead provide means to formulate queries on the semantic, data model level:

> "In this way, Semantic Web applications have to specify in a high-level language only *which* resources need to be accessed, leaving the task of determining *how* to efficiently store or access the descriptions to the underlying *RDF database engine*." [KMA+02]

Similar considerations apply to Topic Maps. Semantic query languages are also commonly called "graph-based", as they rely on the graph representation of ontologies.
The main advantage of semantic query languages, apart from syntactical considerations, is their ability to perform certain advanced tasks such as recursively traversing class hierarchies and determining inferred instantiation relationships. This is only possible because graph-based query languages *incorporate the semantics of the underlying data model*, which for instance defines the transitive nature of the subclassOf-relationship. Without this knowledge, only direct instances and subclasses can be determined automatically. Providing these kinds of "basic inference" is however crucial for user interaction purposes: since the query author is expected to think in terms of the data model, query results must live up to obvious expectations: if a query author wants to retrieve all instances of a certain class, all instances of all subclasses must also be included in the answer, if complete abstraction from the actual implementation should be achieved. If more powerful methods of inference are needed, dedicated *inference engines* can be used; see Section 4.3. A formal approach of performing queries on DAML+OIL ontologies using inference is for instance presented by Horrocks and Tessaris [HT02].
For ontologies expressed with RDF, another, more light-weight method to perform queries exists. Due to the simplicity of the RDF data model, which basically consists of nodes and directed arcs, RDF ontologies can be described by 3-valued statements (triples), which are usually persisted in *triple stores* as outlined in Section 4.1.2. The triple notation of RDF ontologies has been used as a basis for a number of query languages which essentially allow for selecting triples based on filter expressions. Although triple-based query languages do not provide actual abstraction from the implementation (the triple store), they obscure many implementation details such as special treatment of literals, redundancy for the sake of performance etc. The benefit of triple-based languages is that they can easily be implemented (at least in triple stores) and are usually well supported by the storage modules themselves: for instance, rdfDB [Guh] and 3store [RTS] use dedicated index files that allow for fast query execution based on triple filtering using URI hashes.
Topic Maps, on the other hand, are not subject to triple-based languages or similar simplifications, because their data model can not be generalized any more (at least no other standardized versions exist). This is also the reason for Topic Maps query languages always fully incorporating the semantics of the Topic Maps data model.

### 4.2.1 RDF Query Languages

In this section, popular query languages for RDF are presented and their most important features are described.

**RQL:** RQL [KMA+02] is one of the few graph-based query languages for RDF. The specification of RQL introduces the "Formal Data Model", a semistructured

type system that is used by RQL in addition to the RDF and RDF Schema standards. The Formal Data Model clearly distinguishes between data, schema and metaschema layers within an ontology and also imposes some additional constraints on the original standards (such as prohibiting loops in hierarchies). It is the foundation of the RQL query language specification itself: "Exploiting its formal background, *RQL* constitutes a typed, declarative query language for uniformly navigating on RDF/S graphs at all abstraction layers" [KMA+02].

RQL uses predefined functions such as `subClassOf` and `subPropertyOf` to traverse recursively the class and property hierarchies in the ontology. Some of these functions are closely related to the according vocabulary provided by RDF/S, others like `leafclass` are provided for convenience. Answer sets can be limited to direct predecessors and successors in the hierarchy by using the "`^`" operator. Binary operators like `=`, `<`, `>` etc. can also be used to compare hierarchical relationships between classes (`Painter < Artist`). Projection[5] is provided by a SQL-like syntax, as are the set operations `UNION`, `INTERSECT` and `MINUS`.

One of the most interesting aspects of RQL is its *functional nature*: RQL operates on arbitrary RDF structures (graphs), and all query answers are again returned as "bags" or "sequences" as defined by the RDF standard. This makes it possible to use the output of a RQL query as input of another, a process which is called *functional composition*[6]. The possibility of "chaining" RQL queries significantly increases RQL's expressive power.

**RDQL:** The "RDF Data Query Language" [Sea04a] is a quite simple language derived from the discontinued SquishQL query language and used by a number of RDF systems. RDQL is triple-based, using triples with constant values and variables as selectors:

> "An RDQL consists of a graph pattern, expressed as a list of triple patterns. Each triple pattern is comprised of named variables and RDF values (URIs and literals). An RDQL query can additionally have a set of constraints on the values of those variables, and a list of the variables required in the answer set." [Sea04a]

RDQL uses a SQL-like syntax to allow for projection. No recursive traversal of hierarchies is provided, and the language is not functional (it uses triples as input and provides variable bindings as output).

**SeRQL:** This graph-based query language is used in the Sesame RDF framework and is based mostly on RQL. It inherits RQL's ability to traverse class hierarchies

[5] Projection allows for including only relevant variables in the final answer set, thus improving performance by removing all temporary variables.

[6] For instance, SQL is also a functional language, which can use subqueries as input for other queries.

and offers similar built-in functions. SeRQL is also able to query for reified statements, a functionality that is missing in RQL.

Unlike RQL, SeRQL offers two different methods for retrieving query results: either the answer is in tabular form, binding query variables to values (using the `SELECT` clause similar to SQL), or a new RDF graph can be constructed from the values determined by the query (using the `CONSTRUCT` clause). The latter method "returns a true RDF graph, which can be a subgraph of the graph being queried, or a graph containing information that is derived from it" [SeR]. Construct clauses would enable functional composition of SeRQL queries, but no nesting of queries is supported at the time of writing.

**SPARQL:** The SPARQL query language [Sea04b] is being developed by the "W3C RDF Data Access Working Group" and is based on the triple-model. It borrows many concepts from RDQL, but adds functionality to construct output graphs similar to SeRQL. Since no actual query nesting is supported, functional composition of queries is again not possible.

Since SPARQL does not incorporate any predefined RDF/S vocabulary, it lacks any support for inference: "A query processor is unaware of any inference an RDF store may provide and SPARQL makes no distinction between inferred triples and asserted triples" [Sea04b]. It should be noted that SPARQL is still a W3C Public Working Draft at the time of writing. Also, SPARQL may become increasingly interesting for remote ontology querying purposes once the definition of the SPARQL remote access protocol is published.

**TRIPLE:** TRIPLE [SD02] is a query language based on Horn-logic and F-Logic (see Section 4.3.4) that focuses on RDF inference and transformation tasks. Although basically operating on triples (hence the name), its expressiveness can be considered to be very powerful, mainly due to the layered nature of the language. In contrast to other query languages, TRIPLE does not predefine certain predicates to support the semantics of one representation format (like RDF/S or Topic Maps), but provides means for *defining rules*. This makes it possible to use TRIPLE for a variety of languages, including RDF/S, OWL, Topic Maps and UML; it has been used for RDF-based languages mostly.

As Horn rules are not powerful enough to represent the complete semantics of languages like OWL, external reasoning programs (DL classifiers) can be plugged in to support or replace the built-in inference engine. TRIPLE is also able to deal with reified statements and allows for defining rules with full First Order Logic syntax. Skolem functions can be used for ontology mapping and integration purposes. TRIPLE offers both a ASCII-based syntax and a RDF-based syntax intended for human or machine consumption, respectively.

**Versa:** The graph-based query language Versa allows for easy navigation along the arcs in a RDF graph. Its main strength is the support for both forward and backward traversals of paths within a RDF model: in forward traversals, navigation occurs from subject to object, whereas in backward traversals, the object is the starting point. Traversals can also be done trasitively, thus supporting the transitive nature of some relationship types (like `subClassOf`).

Versa makes heavy use of functions, which gives it a LISP-like appearance. In addition to functions specific to the RDF standard (like `all()`, which retrieves all resources from a model), a large number of string, boolean, logic and set functions is provided by the language.

This is of course not an exhaustive list of all RDF query languages available; however, some query languages were either developed for theoretical purposes only and were never implemented for actual use, or are still in a very experimental state and thus not appropriate for "real world"-usage purposes.

### 4.2.2 Topic Maps Query Languages

This section lists most of the available Topic Maps query languages together with their respective characteristics.

**tolog:** The tolog query language [Gar03b, Gar04b] combines elements from Datalog (a subset of Prolog) and SQL to answer queries on Topic Maps. At the time of writing, tolog is the most complete query language available for Topic Maps, with several real world implementations actively using it. Therefore, it is also considered to be the most promising candidate for the upcoming TMQL standard (see below).

Tolog uses built-in predicates such as `topic()`, `association()`, `instance-of()`, `type()`[7] etc. for modelling the according semantics of the Topic Maps standard. Answers are always in tabular form; no functional composition of queries is possible.

Associations in a Topic Map can be used as "dynamic predicates":

```
composed-by(puccini :  composer, $OPERA : opera)?
```

will for instance retrieve all operas composed by Puccini in a Topic Map[8]. Since Topic Maps associations are n-ary and undirected, the order of the values in a predicate is meaningless to the query processor. Therefore, values must be

[7] The predicate `type` models subclass relationships.
[8] The complete example queries for tolog can be found in [Gar04b].

followed by the type of role they play in the association (i.e. "composer" and "opera" in the example).

Similarly, occurrences can be used as dynamic predicates using a syntax like `date-of-birth($COMPOSER, $DATE)?` which retrieves all composers (topics, actually) and their birth dates in the Topic Map. Occurence predicates are always binary, with the first value representing the topic and the second value the occurrence.

Other tolog features are the support for boolean operators, sorting, counting, projection of output variables and "non-failing clauses" (i.e. "a clause that will produce a value for a variable if it can, but if it cannot still won't cause the query to fail" [Gar04b]).

One of the most interesting features of tolog is its ability to represent user-defined inference rules which are executed and evaluated by the query processor whenever a query is to be answered. Inference rules are specified like predicates, taking an arbitrary number of variables as parameters and defining a function body which uses the input variables for other predicates:

```
influenced-by($A, $B) :- {
  pupil-of($A : pupil, $B : teacher) |
  composed-by($OPERA : opera, $A : composer),
  based-on($OPERA : result, $WORK : source),
  written-by($WORK : work, $B : writer)
}.
```

Inference rules can use other inference rules and may also be recursive, which makes it possible to transitively traverse Topic Maps. They can be added to a query directly, or they may be stored in external files and imported via the `import` statement whenever needed.

**Toma:** This query language [Pin04] also supports the semantics of the Topic Maps data model through built-in functions like `$topic.type()`, `$topic.super()` etc. Associations are modelled using the `->` operator. Projection and set operations are also present, but many other features expected from a query language are still missing. Among them are the standard comparison operators (`<`, `>` etc.), ordering and grouping of results (similar to SQL), and creating user-defined functions (analoguous to the inference rules in tolog).

**AsTMa?:** The AsTma? query language [Bar03a, Bar03b] is part of the AsTMa* language family and uses AsTMa= and AsTMa! for notation and specifying constraints respectively. Topic Maps semantics are again incorporated as built-in keywords, enabling the search for topics, associations, occurences etc. Queries

can be wrapped as functions which can be used by other functions, thus encapsulating the complexity of queries and allowing for elegant combination of queries. Functions can return simple values, lists, XML markup or entire Topic Maps.

**TMPath:** TMPath [Bog04] is actually not a complete query language, but can be used to address parts of a Topic Map in XPath-like style. It relies on the TMDM and incorporates its semantics through according keywords ("steps"). Toma allows for quantified and conditional expressions; transitive assciations are supported through the `//` operator.

**TMQL:** All Topic Map query languages presented so far are not formally standardized, which means that at the time of writing no official standard for querying Topic Maps exists. To overcome this situation, the Topic Map Query Language is being developed by the ISO/IEC JTC1 SC34 WG3 and will eventually become official standard ISO 18048. Although the need for a standardized query language has been recognized already several years ago, no specific draft (apart from a very early "clay man version"[9]) of the language exists yet. The reason for this was the lack of a formal data model for Topic Maps which would enable a precise definition of the query language. The revised ISO 13250 standard now also includes the "Topic Map Data Model" (see Section 3.1.9) which was finished in early 2005 but has no official status yet. Workings on improved versions of the TMQL are expected to start soon.

Some preliminary considerations on the requirements [GB03a] and use cases [GB03b] of the TMQL already exist, together with a few proposals on the syntax of the language [Ksi00]. Also, many popular query languages (tolog, Toma, AsTMa?, TMPath, XTMPath [BG02] etc.) are regarded as candidates for borrowing syntactical constructs to the TMQL; the final language definition is likely to include elements from various languages. The clay man proposal uses a SQL-like syntax mostly influenced by tolog, with added capabilities for path-based navigation, both built-in and user-defined functions, querying multiple Topic Maps, specifying existential and universal quantifiers as well as supporting several auxiliary functions, such as constructors for answers in Topic Maps format (e.g. which Topic Map elements should be included in the query response).

Due to the lack of an official standard, many Topic Map frameworks either support one of the languages presented above or enable access to a Topic Map by providing according APIs, which generally not only allow for retrieving parts of Topic Maps, but also for inserting new items and updating existing ones.

[9] Presentation slides covering the functionality of the clay man version can be found at `http://www.jtc1sc34.org/repository/0502.pdf`.

Again, some more query language for Topic Maps exist, but are not described here in detail because of their incomplete implementations.

### 4.2.3 Query Languages Summary

As outlined in the previous section, numerous approaches to semantic query languages for ontologies exist. Some of them are specified in detail, while others are still of experimental nature. Common tasks such as finding direct instances of classes are usually supported well, whereas advanced properties like functional composition or transitive traversals are not implemented in many languages. Since the latter is usually considered to belong to the "inference part" of an ontology, this is not necessarly a disadvantage, even more as many languages try to be as simple as possible. Generally, it remains an open question to which extent inference capabilities should be incorporated into query languages; reasoning with ontologies is discussed in detail in Section 4.3.

With respect to traditional query languages like SQL, many query languages appear to be rather poorly equipped with built-in functions. Ordering, grouping and set operations on result sets are commonly used features of SQL, but are often not present, as sometimes are boolean and comparison operators. Aggregate functions are also implemented rarely.

Several approaches exist regarding the input and output formats of ontology query languages. In most cases, variables are specified in some kind of "pattern" and are then bound to actual values by the query processor, resulting in a tabular answer set comparable to a traditional SQL result. Some languages allow for creating triples or even graphs from the variable bindings using a "construct"-pattern, which may be a very interesting feature for many applications and allows for functional composition of queries.

One more observation should be noted here: none of the presented query languages allows for modifying the underlying ontology, which means that updates to the knowledge base must be performed by calling a separate API. While performing updates is not a strict requirement for query languages, it may still be a desirable feature which is for instance recognized by the developers of the TMQL: "it is agreed that part 1 of TMQL will cover retrieval scenarios only; updating TM stores is left for part 2, which will be handled separately" [TMQ].

## 4.3 Reasoning with Ontologies

The original definition of *ontology* given in Section 1.1.1 emphasizes the fact that ontologies are not only about concepts, but also about the relationships between them. Actually, the *explicit existence* of relationships between concepts is an outstanding

feature of ontologies, compared to traditional systems dealing with information representation (such as databases) where relationships are only represented implicitly by primary or foreign key columns. It allows for a feature that is conceptually unavailable[10] for database systems, namely *automated reasoning* over the concepts and relationships in the ontology.

There are two main application areas for automated reasoning within ontologies: performing *semantic validation* of concepts, and *inferring implicit knowledge* from explicit facts using rules. Semantic validation focuses on the correct "content" of concepts and their properties, in contrast to syntactic validation which evaluates the correspondency of an ontology with XML and RDF standards. RDFS and OWL provide constructs to constrain the use of concepts and properties (e.g. by specifying values for `domain` and `range` keywords), and ontology data can be checked against these constraints using an *inference engine*:

> "Reasoning is important to ensure the quality of an ontology. It can be employed in different development phases. During ontology design, it can be used to test whether concepts are non-contradictory and to derive implied relations. In particular, one usually wants to compute the concept hierarchy. Information on which concept is a specialization of another and which concepts are synonyms can be used in the design phase to test whether the concept definitions in the ontology have the intended consequences or not." [BHS03]

However, performing semantic validation can be considered to be a side-effect of the true purpose of an inference engine, which is to *derive implicit information* from the explicit facts contained in an ontology. The most interesting observation is that this process of deducing new knowledge can be performed automatically without any user interaction (and thus without additional information provided by a human operator). It is actually a bit misleading to speak of "new knowledge", since all information is indeed contained in the existing facts, but the process of deducing new knowledge is similar to the way human reflection works. For example, the brother of any person's parent is said to be an uncle of that person. If it is known that a person A has a father B, and that C is the brother of B (these are the explicit facts), we can derive that C is the uncle of A (which is derived knowledge). Similary, if the concept "cow" is known to be instance of the concept "mammal", and "mammal" is known to be a subclass of the concept "animal", the conclusion is that cows are also animals (which is a quite obvious, but originally not directly available fact).

[10] One of the reasons for this is that, in contrast to ontologies, databases are not aware of their respective schemas, as outlined in Section 2.5.

### 4.3.1 Logics in the Semantic Web

From a developer's point of view, dealing with inference is not a straightforward task, as a firm background in formal logics is needed. This is also the reason for many ontology frameworks not supporting inference mechnisms at all, or only providing basic functionality. It should be noted that inference is indeed not needed by many applications which only require simple representation of explicit concepts and relationships. On the other hand, the vision of the Semantic Web includes agents that autonomously act on behalf of their owners. But even to achieve very simple goals, such agents will have to revert to a knowledge base representing human common sense, which is very likely to be too large and complex to be populated manually with explicit facts only. Apart from being infeasible, such an approach would also thwart the original purpose of an ontology which is to create clear, unambiguous hierarchies and relationships between concepts. Also, in order to be autonomous, an agent must be able to cope with incomplete facts and remote ontologies, effectively compensating the lack of information by mapping new assertions into its local knowledge base and performing inference to deduce the knowledge that is needed to accomplish the current task. For instance, a travel agency may provide assertions about departure dates and times of flights, but not a complete ontology about transportation. An agent that attempts to book a flight for its owner must therefore be able to recognize that airplanes are a means of transportation and adequate for travelling long distances, and that oil-tankers are not (although they also have departure dates and times).

Within the Semantic Web layered architecture depicted in Figure 1.1, inference may occurr on two layers: the "Ontology vocabulary" layer may provide inference for subsumption and instance relationships (as Description Logics do, see next section), whereas other rules are represented by the "Logic" layer (e.g. Horn-logic rules).

Inference within ontologies is not a stand-alone feature, but is often tightly integrated in storage and/or query facilities. Two different approaches exist for enabling inference within ontology frameworks: the *data-driven* approach and the *hypothesis-driven* approach. The former, also called *forward-chaining*, starts the inference engine whenever new (explicit) assertions are to be inserted in the ontology. Implicit facts are then deduced by the inference engine and also stored in the ontology. This "exhaustive forward inferencing" is generally considered to be unfeasible for larger ontologies due to its greatly increased need for storage space (see e.g. [Las02]), but as Broekstra and Kampmann show [BK03], it is well suited for flat ontologies of medium size. Special attention must be given to statement deletion, which requires all deduced facts to be revised. This can either be achieved by marking inferred assertions upon insertion, removing them and recomputing the deductive closure of the ontology (the "naive approach"), or by establishing a sophisticated truth maintenance system which tracks the logical dependencies between assertions. The advantage of forward inferencing is that queries can be answered very fast, as no additional inference must be performed

to retrieve answers from the ontology.

The hyptothesis-driven approach, also called *backward-chaining*, performs inference only when needed to answer queries. No truth-maintenance mechanisms are needed, as the computed results are used for query answering only and are not stored in the ontology itself (of course, this behavior is usually modified in favor of a caching mechanism to avoid multiple computation of the same implicit facts). "The main advantages of such an approach are the decrease in required storage size and upload time, while the main disadvantage is the decrease in performance of query answering" [BK03].

### 4.3.2 Formal Foundations

Ontologies as organizing principles for knowledge management have been studied by the artificial intelligence and knowledge representation communities for several decades, using *formal logic* as foundational theory for implementing automated reasoning mechanisms. In fact, logic models have been found to be closely related to ontologies: the *knowledge bases* used by logic languages are comparable to the information contained in an ontology (e.g. represented by RDF/OWL or Topic Maps), although such knowledge bases usually use somewhat different representation syntaxes (such as KIF, or notations closely related to the abstract syntax used in formal logic) for expressing facts about and relationships between concepts.

Many different "kinds" of logics were developed since the beginning of the last century, with various degrees of complexity and expressiveness. Research has mostly concentrated on *first order logic*[11] and various subsets thereof, such as *Description Logics* and *Horn-logic*. The reason for this is that while *higher order logics* are more expressive than first order logics, they are hard to reason with, because they do not admit a proof theory (this is essentially a corollary of Gödel's incompleteness theorem): in higher order logics (e.g. second order logics), there are true sentences which can not be proven true. This makes higher order languages unsuited for most applications beyond theoretic research.

First order logic, on the other hand, is semi-decidable, which means that for every formula either the formula itself or its negation can be proven true in finite time. Although this makes a big difference from a theoretical point of view, in practice inference engines (so called "theorem provers") for full first order logics are unable to scale with large sets of data. Hence, most actual implementations of inference engines focus on *subsets of first order logic*, mostly *Description Logics* and *Horn-logic*.

[11] Roughly spoken, first order logics can not quantify over predicates, in contrast to higher order logics.

### 4.3.3 Description Logics

Description Logics are a family of languages of varying complexity and computational properties that can be used to represent knowledge. Description Logics focus on the representation of concepts (or classes); the basic idea is to use constructors to build complex classes from simpler concepts. "The name description logics is motivated by the fact that [...] the important notions of the domain are described by concept descriptions, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL" [BHS03]. Description Logics contain a *terminological* formalism ("TBox") for defining hierarchies of concepts, and an *assertional* formalism ("ABox") to state properties of instances (also called individuals).
Special attention is given to the computational behavior of description logic languages:

> "In order to ensure a reasonable and predictable behavior of a DL system, [...] inference problems should at least be decidable for the DL employed by the system, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressiveness of DLs and the complexity of their inference problems has been one of the most important issues in DL research." [BHS03]

As first order logic is only semi-decidable (and thus often considered to be undecidable for practical use), Description Logic languages must be designed very carefully to maintain decidability. This can be achieved by allowing only constructors that avoid to make a language undecidable, and avoiding constructors that do so. Of course, this has negative effects on the expressiveness of a Description Logics language. Summarized it can be said that designing a Description Logics language is always a tradeoff between allowing for expressiveness and still maintaining decidability.
Some of the most prominent Description Logics for ontologies are $\mathcal{SHIQ}$ [HST99] and $\mathcal{SHOQ}(\mathbf{D})$ [HS01], both members of the $\mathcal{SH}$ family of Description Logics whose constructors and axioms "include the boolean connectives (intersection, union and complement), restrictions on properties, transitive properties and a property hierarchy" [HPSvH03] and thus support many of the features needed by ontologies. In addition to these properties, $\mathcal{SHIQ}$ also uses inverse properties and generalized cardinality restrictions, whereas $\mathcal{SHOQ}(\mathbf{D})$ "adds the ability to define a class by enumerating its instances[12] [...] and support for datatypes and values (e.g., integer and string datatypes, and values such as '35')" [HPSvH03]. Both $\mathcal{SHIQ}$ and $\mathcal{SHOQ}(\mathbf{D})$

[12] This is also known as *existential* definition of classes.

are decidable and of worst-case deterministic exponential time (ExpTime) complexity, which makes them well suited for reasoning in ontologies.
Actually, Description Logics and OWL are closely related, since the predecessor of OWL, DAML+OIL, was already heavily influenced by Description Logics research.

> "Description Logics, and insights from Description Logic research, had a strong influence on the design of OWL, particularly on the formalisation of the semantics, the choice of language constructors, and the integration of datatypes and data values. In fact OWL DL and OWL Lite [...] can be viewed as expressive Description Logics, with an ontology being equivalent to a Description Logic knowledge base. [HPSvH03]

While OWL full is undecidable (and hence difficult to use for inference engines), OWL DL can be mapped to the expressive $\mathcal{SHOIN}(\mathbf{D})$ Description Logic, which is decidable and of worst-case non-deterministic exponential time (NExpTime) complexity. $\mathcal{SHOIN}(\mathbf{D})$ is an extension of $\mathcal{SHOQ}(\mathbf{D})$, adding inverse roles but being restricted to unqualified number restrictions. The NExpTime complexity makes inference possible for OWL DL, but "there is as of yet no known 'practical' complete algorithm for inference in $\mathcal{SHOIN}(\mathbf{D})$, i.e., one that is likely to perform well on the kinds of problem encountered in typical applications" [HPSvH03]. OWL Lite, on the other hand, is comparable to the $\mathcal{SHIF}(\mathbf{D})$ Description Logics, which is known to be of ExpTime complexity, thus making OWL Lite a good candidate for inference services with limited expressiveness.
While Description Logics such as $\mathcal{SHOIN}(\mathbf{D})$ offer a strong formal foundation for ontologies, there are also some limitations to the inference services they provide. Above all, inferencing with the Description Logics described above is always *limited to classification and subsumption* of concepts, hence the name "classifiers" for Description Logics inference engines. This means that a new class can be placed automatically in the hierarchy of concepts by a classifier, and for any new individual (instance), the classes it belongs to can be determined. The two most popular modern classifiers are FaCT [Hor01] and RACER [HM01], which are both able to perform inference on $\mathcal{SHIQ}$ Description Logics knowledge bases. BOR [BOR] is another Description Logics reasoner designed for DAML+OIL and compliant to OWL. It's derivative SeBOR is integrated in the Sesame framework.
Classification and subsumption are certainly some of the most important and frequently used kinds of inference, especially because they allow for semantic validation (and thus always produce sound and complete results). There are however cases where additional functionality is needed. An important limitation of (decidable) Description Logics is the lack of *property chaining*, i.e. the possibility of defining *rules* that are globally valid and not tied to a particular instance. This makes it impossible to create rules such as "an uncle is a parent's brother" and to reason with such definitions. Property chaining was excluded deliberately from languages such as OWL, because

"allowing relationships to be asserted between property chains [...] would make OWL entailment undecidable" [HPSvH03]. Other limitations are discussed in Chapter 5.

### 4.3.4 Horn-Logic

Horn-logic is another subset of first order logic which is restricted to definite clauses, i.e. clauses with one positive literal. Horn-logic and its most prominent language Datalog[13] have been used for inference in deductive databases and are now considered as interesting candidates for providing additional inference support for ontologies. Ontologies based on Description Logics are only able to represent concepts and relationships between them (the so called *extensional knowledge*), but are unable to represent rules (*intensional* knowledge). Horn-Logic based languages can be used to express both extensional and intensional knowledge. While Horn-Logic itself is undecidable, certain subsets obtained by applying restrictions (as for instance imposed by Datalog) are decidable.
In contrast to Description Logics, which primarily deal with consistency checking and classification, Horn-logics focus on deducing new knowledge from existing one by using rules. These rules consist of a head containing the assumptions, and a body containing the facts to be deduced if the assumptions can be proven to be true. Rules may also be recursive, thus greatly increasing the expressiveness of Horn-logic languages.
Due to its background in the database area, Datalog assumes a relational data model which is not very well suited for supporting ontologies. The newer *F-Logic*[14] language [KLW95], which is also based on Horn-logic, adopts a object-oriented data model and was also inspired by frame-based languages. Several implementations of this language exist, such as XSB [XSB], Flora-2 [YKZ03, CFJ03], F-OWL [ZFC04] and OntoBroker [FDES98]. These systems have shown to be highly scalable, also due to the fact that they usually use the *closed world assumption* ("negation by failure") like traditional database systems do. Description Logics, on the other hand, are said to be less scalable for very large knowledge bases; they use a *open world assumption*, which, despite being closer to reality[15], is the reason for many problems commonly found in Description Logics.
Recent approaches to ontology modelling accept the strengths and weaknesses of Description Logics and Horn-logic based languages and try to combine the best of both worlds. A prominent example is the TRIPLE language presented in Section 4.2.1, which is based on Horn-logic and F-Logic and may call (external) Description Logics classifiers if needed (hence the term *hybrid language*). Integration of Horn-logics and Description Logics is currently an active area of research in the Semantic Web

[13] Horn-logic with 0-ary function symbols only
[14] "F" stands for "Frames"
[15] After all, an ontology is always only a partial representation of all possible things that exist; therefore, the non-existence of concepts not contained in an ontology can ususally not be assumed.

community (see e.g. the RuleML initiative [RML]).

### 4.3.5 Reasoning with Topic Maps

Inference engines supporting Topic Maps are very rare. The reason for this can probably be found in the formal foundation of Topic Maps, whose design was not influenced by Description Logics as it is the case with OWL. Therefore, properties of associations like transitivity are not present in the Topic Map standard[16]. As Newcomb and Biezunski state, "the whole question of inference rules is not addressed in ISO 13250; inferencing is regarded as a property of systems and not of documents, and therefore inference rules are regarded as outside the scope of the standard" [NB00]. In fact, considering the n-ary and omnidirectional nature of Topic Map associations, properties like transitivity can not be applied directly (as they are meaningful for binary relationships only). Subsumption and classification algorithms can of course be applied to Topic Maps, since corresponding constructs exist (see Section 3.1.4).
In fact, the only way of reasoning with Topic Maps is provided by the query language tolog (see Section 4.2.2), which allows for creating rules similar to those used by F-Logic. The drawbacks of such a solution are twofold: first, rules must be specfied using a syntax different from the standard XTM syntax. Second, they are not part of the Topic Map itself, but are stored in external files. Also, the rules are always executed by the query processor and the results can not easily be stored in the Topic Map, which means that only backward-chaining inference can be performed. Any caching strategy has to be implemented by the query processor, too. Maybe some of these issues will be remedied with the upcoming TMQL standard.

### 4.3.6 Summary

One of the key purposes of ontologies is to provide meaningful (semantically enriched) data for both computer agents and humans. The existence of richly populated ontologies representing some more or less sophisticated level of "common sense" is an important requirement for the realisation of autonomously acting agents. An agent without a knowledgebase that enables it to act reasonably will never be able to carry out useful tasks. However, such rich ontologies need a lot of manual input from human ontology modellers (as demonstrated by the Cyc ontology); yet it is impossible to provide a complete knowledge base of common sense in this way, which demonstrates the need for autmatically deducing implicit facts from explicit information.
The formal foundations of ontologies and especially the close relationship between Description Logics and ontology representation formats such as OWL offer a good basis

[16] In the early days of Topic Map standardization, properties of associations were obviously considered to be included in the standard; see e.g. [Rat99].

for automated reasoning. Although different approaches to inference had been investigated for several decades especially among artificial intelligence researchers, the recent wave of interest in ontologies caused by the Semantic Web initiative encouraged the development of new, efficient inference engines for both Description Logics classifiers and F-Logic reasoners. While there are still many issues to be resolved, integrating classification and rule-based inference is a promising approach towards inference engines that are not only interesting from a researches point of view, but indeed ready for use in real world applications.

# 5 Conceptual Issues

After presenting basic concepts of ontological modelling in Chapter 2, introducing RDF, OWL and Topic Maps as XML-based representation formats (Chapter 3) and describing some of the partially solved issues emerging whenever applications using ontologies are to be implemented (Chapter 4), this final chapter is dedicated to some of the most unclear, yet very important aspects of ontologies, especially in the context of the Semantic Web. While some of these issues are related to ontologies in general, others are rather specific properties of the OWL or Topic Map representations formats. Both of them must be considered to be equally important, since OWL and Topic Maps can be considered as the only languages that will potentially be adopted on a large scale.
A common property of the problems presented in the following sections is that they can not be solved by *gradually* improving certain aspects of the underlying systems or technologies, but must be seen as *conceptual weaknesses*. In this respect, they are far more critical than the challenges described earlier, like providing scalable storage for ontologies (see Section 4.1), as those can be improved by applying better algorithms, faster hardware etc.
The Semantic Web initiative focuses on autonomous interpretation of data by agent-based systems, a vision that imlicitly requires agents to revert to some kind of common sense knowledge. Ontologies, which shall constitute the primary knowledge containers for agents (and humans as well), must therefore be able to represent such common sense knowledge, a requirement which turns out to be unequally harder to meet than originally assumed by many ontology engineers.

## 5.1 The Problematic Notion of Identity

### 5.1.1 Problem Description

Ontologies primarily focus on representing "things", or concepts, by creating electronic proxies ("binding points") for them in order to allow for linking arbitrary assertions to them. Since each concept can only have exactly one proxy within a single ontology, it is a crucial task to determine the equality (or inequality) of any two concepts. Generally, this is referred to as "establishing identity" for a certain concept, which is crucial only at first sight. In fact, the notion of identity is rather a philosophical problem, but nevertheless immediately comprehensible for anybody: while it is obvious that identity

is not directly linked to one specific property of a concept but rather to the sum of all its properties, it still remains a difficult question to what extent properties can be modified without changing the identity of the respective concept.
Despite this fundamental problem, ontology representation formats such as RDF/OWL and Topic Maps have adopted quite a simple model which assumes the existence of some kind of unambiguously determinable identity for every concept, whether it is an object or living being of the real world, an abstract concept, an electronic resource, or a class of such concepts (which is esentially seen as an abstract concept itself). Although this simple solution is technically feasible, it must be considered to be rather superficial and turns out to be no real solution whenever different points of view of the same concept are to be represented, as discussed in Section 5.2.
Also, it should be noted that the existence (which is yet another quite unclear term by itself) of a concept, especially if talking about things in the real world, must not be confused with its *lifetime*. The ISO 13250 Topic Map Standard explicitly states that proxies may refer to inexistent concepts as well:

> "In the most generic sense, a 'subject' is any thing whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever." [BBN02]

The RDF standard, on the other hand, resorts to the definition of the term "resource" in RFC 2396, which only mentions that a "resource can be anything that has identity" [BLFIM98]. This is considered to be a rather problematic definition (see also Section 2.1); after all, inexistent concepts may still be present as abstract ones in the mind of humans, e.g. the former Soviet Union etc., and therefore an ontology may certainly refer to them through an electronic proxy.
Apart from the difficult philosophical aspects of "identity", the RDF standard also fails to provide a method for unambiguously referring to concepts which are not network-retrievable. This has already been discussed in Section 2.3.3, but is mentioned again here because at time time of writing no possible solution has been proposed by the creators of the RDF standard, thus often creating confusion among the adopters of the new standards.
Topic Maps, on the other hand, carefully distinguish between network-retrievable resources (which are referred to by Subject Addresses) and other concepts (which are referred to by Subject Indicators). Interoperability between ontologies is achieved by creating Public Subject Indicators (PSIs, see Section 2.3.1), a process that is assumed to be open and distributed, since anybody can create Published Subjects at will:

> "Anyone can publish PSIs, from the largest international organizations to communities of interest, enterprises and even individuals. There is no approval process and no registration authority. The adoption of PSIs can therefore be an open, bottom-up, and distributed process." [Pep03]

It is however questionable whether this process will actually work out. One criticism of the method described above is that interoperability of ontologies can only be guaranteed if the *same PSIs* are used for the concepts that are represented (again, determining *equality of concepts* is subject to the considerations made earlier in this section). At least, explicit mappings must be provided for different PSIs referring to the same concept in order to achieve integration of ontologies. Creating such mappings is not only a laborious task and possibly of little interest to the publisher[1], but may also be objected on purpose, e.g. in a competitive scenario involving several publishers of the same domain. While being an interesting suggestion to the problem of interoperability among ontologies, PSIs yet have to prove their applicability under real circumstances.

### 5.1.2 Possible Solutions

The philosophical problem of establishing identity is very unlikely to be solved by means of technical standards, if a single "solution" can be assumed to exist at all. With respect to the scope of this thesis, only the deficits of the RDF standard and possible workarounds will be discussed.
As outlined in the previous section, the Topic Map standard already includes a clean separation between network-retrievable subjects and other concepts that can not be referred to directly by machines. Thus, a modification of the RDF standard in order to allow for a similar distinction would be the logical consequence, albeit the most unlikely one, as existing standards are not subject to changes. Such modifications have been proposed by several people, as outlined in the paper "Curing the Web's Identity Crisis":

> "Others in the Web community have made more useful proposals, including Sandro Hawke, who suggests that the dual use of URIs be formally recognized in RDF. Hawke uses the terms 'page-mode' and 'subject-mode' to make exactly the same distinction as that made in Topic Maps between subject addresses and subject identifiers. [...] First of all, the RDF model [...] must be adjusted to recognize the distinction between information resources and 'things in general' (i.e., between addressable subjects and arbitrary subjects)." [PS03]

Two possibilities exist to introduce such a modification:

1. The RDF standard is extended to include a special vocabulary for making the necessary distinction described above. This is also the approach taken by Topic Maps with the `resourceRef` and `subjectIndicatorRef` elements and is certainly the most radical and elegant way of dealing with the problem.

[1] In this case, the term "publisher" refers to publishers of "Published Subjects".

2. The syntax of the URI referring to a concept is modified. In fact, the RDF and OWL standards and also many examples use the #-sign in URIs to indicate that an URI is not referring to a network-retrievable resource; however, no official standard or convention exists: "Throughout RDF, strings like '`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`' are used with no consistent explanation of how they relate to the web" [Haw03]. While using URIs including the #-sign to indicate an arbitrary subject would not require to change the RDF standard itself, this approach neglects the fact that the #-sign is already used as fragment identifier in URIs:

   > "But using '#' like this, while perhaps reasonable in RDF, is not in keeping with the general architecture of the web. Both subject-mode and page-mode identification are useful for both full pages and for fragments. In HTML, it's common to point to parts of other documents [...]. It's also common to use full-page addresses to identify a subject [...]." [Haw03]

   Summarized, the syntactical approach to resolve the RDF problem must be considered as easy to implement, but not universally applicable and is thus ill-suited for general adoption.

In addition to these two options, there is also a third possibility for distinguishing concepts that are network-retrievable and concepts that are not. The RDF Schema standard provides the built-in `rdfs:subClassOf` predicate to allow for subsumption of concept classes, thus creating a class hierarchy. All RDF classes must be (implicit or explicit) subclasses of the top-most class `rdfs:Class`. This definition could be extended on a conventional basis by introducing two arbitrarily named disjoint classes, e.g. "RealWorldObject" and "WebResource", and relate all other classes to these classes via the `rdfs:subClassOf` predicate. Any RDF instance concept would be further required to be an instance of either "RealWorldObject" or "WebResource" (but not both, as they are disjoint classes), or of one of their subclasses. This has the following effects in practice: any class is either a subclass of "RealWorldObject" (thus representing a class of real world objects) or a subclass of "WebResource" (thus representing a class of web resources), and any instance is either a real world object or a web resource (i.e. linked to one of these classes through the `rdf:type` predicate). Although the proposal presented above is both easy to implement and does not require the RDF standard to be modified, it has no official backup at the time of writing. It is however interesting to note that the developers of the query language RQL use this model even for a relatively small example ontology [KMA+02]. Another drawback is that the proposal is based on a convention only, thus possibly compromising interoperability among arbitrary ontologies that do not adopt the convention.

## 5.2 The Absence of Context and Contextual Constraints

### 5.2.1 Problem Description

One of the most surprising facts about OWL and, to a large extend, also about Topic Maps is that there are no built-in methods for dealing with *contexts* in a clean way. A context is often defined as a set of assumptions and facts that together build a basis for other assertions:

> "A context [...] is a set of assertions, representing a particular set of surrounding circumstances, relevant facts, IF-THEN rules, and background assumptions." [Cyc02a]

Sometimes, contexts are also seen as a specific party's view of a domain, and are thus opposed to the common understanding of an ontology as a *shared set of concepts and assertions* (see also Section 1.1.1):

> "**Ontologies** are *shared* models of some domain that encode a view which is common to a set of different parties;
> **Contexts** are *local* (where *local* is intended here to imply *not shared*) models that encode a party's view of a domain." [BGvH+03]

This point of view is not contradicting the first definition, but emphasizes the different scopes of ontologies and contexts. However, even though ontologies represent shared knowledge, that knowledge can not automatically assumed to be universally valid. In fact, the absence of context in OWL and Topic Maps means that concepts and assertions are not constrained in any way and thus must be considered to be valid under all circumstances. Although the Topic Map standard provides the "scope" element (see Section 3.1.6) for dealing with contextual constraints, its semantics are unclear; also, Topic Maps scopes are far from offering a complete solution to the problem of representing context, as outlined in the next section.
The implications of these findings are far-reaching and problematic especially whenever ontologies should represent common sense knowledge, which tends to be very complex. Ontologies that are unable to deal with contexts reveal serious shortcomings whenever one of the following cases occurs:

**Change of knowledge:** without being linked to any context, concepts and assertions are valid *at all points in time*. Changes to the knowledge base are not only problematic if they collide with "outdated" information (see Section 5.3), but are also retroactive. Thus, many assertions in a knowledge base can only be considered to be true if their validity is constrained to some period of time. For example, woman suffrage was introduced in Austria in 1918; until that point in time, women were not entitled to vote or to be elected. An ontology can only model these facts if their validity is subject to temporal constraints.

**Spatial constraints:** similar considerations apply to the spatial validity of facts in an ontology, as the validity of assertions may be restricted to certain places or locations. The inability of OWL and Topic Map ontologies to cope properly with space and time on a logical[2] level is widely recognized: "[...] spatiotemporal ontologies are in their infancy, in particular because of the lack of an appropriate model, capable of dealing with space and time at the ontological level, and of a suitable reasoning engine" [SCPV04]. Reasoning with *concrete domains* (such as integer numbers) is supported by several inference engines, e.g. RACER, but spatial and temporal inferencing is still unavailable in current systems.

**Different points of view:** whenever different points of view are to be modelled in an ontology, providing relationships to the contexts of the parties involved becomes crucial. Of course one could argue that controversial facts must not be present in an ontology (since it is supposed to contain "shared" information only), but in practice, obtaining an agreement about controversial aspects is usually harder than expected. Therefore, the need for ontologies to represent different points of view of certain concepts and assertions should not be ignored; even the *existence* of certain concepts is not always self-evident. Different points of view can also cause inconsistencies within an ontology; this is discussed in Section 5.3.

The identity of concepts is another critical aspect that may differ depending on who defines a concept. As Doctorow points out in [Doc01], "reasonable people can disagree forever on how to describe something", e.g. "No, I'm not watching cartoons! It's *cultural anthropology*." Since ontologies anticipate the existence of an unambiguous identity, different definitions and descriptions of a concept are not problematic for ontologies, but require descriptive statements (such as the one in the example) to belong to different contexts.

**Fictitious facts:** for obvious reasons, it is very important for any ontology to distinguish between concepts and facts that are believed to be true in the "real world" and those that are considered to be fictitious:

> "In the fictional context of Brain Stoker's *Dracula*, vampires exist; in the standard rational worldview context they don't. Other contexts carve out similar distinguishable eras in time, political or religious points of view, and so forth." [Sto98]

Generally, concepts, assertions and common sense rules that are found in a fictitious context do not apply to real world entities, and vice versa. For instance, a rule could state that all living beings come into existence at some point in time,

[2] This must nut be confused with ontologies *about* space and time; creating ontologies that contain spatial and temporal terms is of course possible and according efforts are being made, but this does not help in constraining the validity of assertions in ontologies in general.

get older over time and eventually die. This is however not true e.g. for most cartoon characters, whose age is not tied to any strict timeline. One could also argue that the whole notion of time is not identical for the real world and many fictitious stories, even though certain aspects such as the monotonic nature of time usually apply to both of them.

In addition to purely fictitious facts, *believes* and *religious assumptions* are also subject to such considerations. Especially with respect to the moral aspects of many religions, *modal logic* (see next section) could be an important addition to the standard logic vocabulary (that is e.g. used by description logic classifiers).

In the next section, we will discuss some possible solutions for these conceptual weaknesses of ontologies.

### 5.2.2 Possible Solutions

#### Temporal and Spatial Contexts

Although in practice it is hard to find assertions that can be considered *universally true* and thus a big need for constraining assertions in this way should exist, neither OWL nor Topic Maps offer a clean solution for this problem. Both representation formats are unable to deal with the continuous nature of time and space, at least at constraint level. This is also true for the *scope* construct in the Topic Map standard, which can only represent discreet constraints of validity.
Summarized it can be said that current ontology representation formats such as OWL and Topic Maps do not include mechanisms for constraining assertions to certain periods in time or certain (geographic) areas. However, they are certainly suited for representing temporal and spatial concepts, such as "interval", "event" etc. Respective ontologies (which are often referred to as *spatial* or *temporal ontologies*) have been created lately, several proposals and examples can be found e.g. in [SCPV04, PH04, CoB04].
An additional step towards proper handling of spatial and temporal properties of assertions would be the creation of inference engines that are able to deal with the temporally and spatially limited validity of assertions. Such inference engines would yield correct results for all queries where such constraints of validity can not be neglected, as in the example given in the previous section. At the time of writing, no such inference engines for OWL or Topic Maps exist.

#### Modelling Points of View

If an ontology has to cope with several different points of view of various parties, two possibilities for doing so have been proposed. The first proposal is to keep concepts and assertions that are "shared" (i.e. common to all parties) in a shared ontology, and

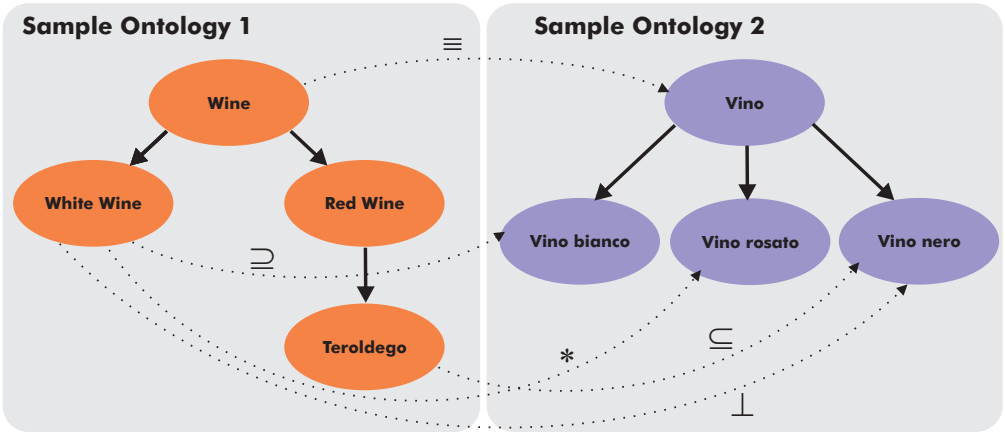


Figure 5.1: C-OWL mappings between two sample ontologies [BGvH+03].

facts that are considered to be "subjective" in separate, local ontologies. The latter ontologies are therefore models of the respective contexts and are thus referred to as *contextual ontologies*:

> "We say that an ontology is contextualized or, also, that it is a *contextual ontology*, when its contents are kept local, and therefore not shared with other ontologies [...]" [BGvH+03]

In order to achieve interoperability between the shared ontology and its contexts, explicit *mappings* are provided that link corresponding concepts:

> "A context mapping allows us to state that a certain property holds between elements of two different ontologies." [BGvH+03]

Mappings consist of serveral unidirectional "bridge rules" which can express different types of relationships between any two concepts [PBZ04], such as "more specific than", "more abstract than", "disjoint from", "compatible with" etc.
An implementation of this idea for OWL ontologies has been presented in [BGvH+03]. Since OWL does not allow for creating mappings to other ontologies (other than a rather simple import mechanism), its syntax was extended, resulting in the *Context OWL*, or *C-OWL*. In Figure 5.1, a sample mapping between two ontologies is illustrated.
The main drawback of this solution is that it is not very feasible in the context of open environments:

> "However, the vocabularies of local ontologies are supposed to be pair-wise disjointed, and the globalization can only be obtained by using explicit

> mappings. It doesn't fit very well in with one of the basic architectural principles of the Web, which allows anyone be able to freely add information about an existing resource using any vocabulary they please." [QG04]

These disadvantages however apply to all solutions that require some kind of manual mapping between ontologies. Therefore, methods for automatically creating mappings have been investigated. In [ES04], (simple) mappings between ontologies are for instance created by calculating the similarity of concepts. Other approaches use Bayesian Networks to improve ontology mappings [MNJ04].
A different approach is followed by the Topic Map standard, which allows for creating *scopes* within a Topic Map ontology. After creating scope elements in a Topic Map, any assertion can be linked to one or more of these scopes, which means that the respective assertion is only valid within the referenced scopes. Assertions that are not "scoped" in any way are universally valid. Although scopes seem to be a good solution for modelling different points of view, they are considered to be problematic in a number of ways. First, scopes can also be used for many other purposes, e.g. for propagating access permissions within an ontology, which can easily lead to confusion since the Topic Map standard does not define any "types of scopes". Second, scopes can not be further organized for instance in a hierarchical manner. This also limits their practical use, as no indirect references to scopes are possible (e.g. for inference purposes).

#### Fictitious Facts

There are many application areas for ontologies where it is crucial to detect and properly "tag" those facts that can not generally assumed to be true. For instance, ontologies have been used successfully for supporting Natural Language Processing tasks (NLP) by providing important background information about concepts for the interpretation process [AKM+03]. Also, the automated population of ontologies from documents such as web pages and text files by using NLP methods is being investigated [CVV04, HSC02, PSL03].
However, the implementations presented so far are not able to detect facts that are only valid in a certain context, which is maybe due to the fact that OWL (which is used for virtually all implementations) is not able to properly represent contexts and associated facts. Yet in reality, a great part of documents that contain written text is either completely fictitious (novels etc.) or contains rhetorical elements such as irony or cynicism which sometimes express facts that must not be taken literally. A promising method for representing contexts, called "microtheories", is used in the Cyc knowledge base. As the primary focus of a microtheory is to maintain consistency, it is described more in detail in the next section.

As already denoted earlier, *modal logic* may be another interesting way to represent uncertain facts, especially in conjunction with moral expressions:

> "Modal logic is, strictly speaking, the study of the deductive behavior of the expressions 'it is necessary that' and 'it is possible that'. However, the term 'modal logic' may be used more broadly for a family of related systems. These include logics for belief, for tense and other temporal expressions, for the deontic (moral) expressions such as 'it is obligatory that' and 'it is permitted that', and many others." [Gar03c]

As modal logic is again a wide area of research by itself, it will not be discussed here in detail. It should be pointed out that some Description Logic reasoners such as FaCT and RACER are able to perform inference in the (weak) "modal logic Km with graded modalities and axioms" [RAC]. However, little additional information on applying modal logic to contextual problems in ontologies is available.

## 5.3 Maintaining Consistency

### 5.3.1 Problem Description

Closely related to the notion of context is the aspect of *consistency* within an ontology. Ontologies are commonly expected to contain only facts that do not contradict each other, thus resulting in a consistent ontology. Especially OWL with its strong background in Description Logics is a good example for an ontology language that requires consistency in order to yield meaningful information. Also, consistency is essential for inference engines, which are unable to deduce implicit knowledge if the knowledge base contains conflicting assertions. With respect to Topic Maps, the lack of a proper ontological vocabulary facilitates the representation of arbitrary facts, even if they contradict each other. However, reasoning is very limited for Topic Maps, which is why we will concentrate on OWL in this section.
Despite of these considerations, it is still surprising that there is no way to properly represent contradictory facts in OWL. This is however necessary if, as already mentioned, different point of views are to be represented, or whenever assertions are tied to a particular context (such as fiction). If we recall one more time the vision of the Semantic Web [BLHL01], ontologies should also contain common sense knowledge in order to properly support automated agents. However, if perfect consistency is required in an ontology, representing complex information is likely to be impossible. The very same observation was made by the creators of CYC:

> "The [...] perhaps most important lesson we learned along the way was that it was foolhardy to try to maintain consistency in one huge flat CYC knowledge base. [...] in the context of working in an office it's socially

> unacceptable to jump up screaming whenever good things happen, while in the context of a football game it's socially unacceptable *not* to." [Sto98]

Such contradictions are not only likely to occurr whenever complex knowledge should be modelled, but may even form the biggest part of the common sense knowledge. After all, there are almost no assertions that can be considered to be true under all circumstances. It is therefore important to account for this fact and provide solutions instead of ignoring it, as it sometimes appears to be the case with OWL ontologies.
Another problematic aspect of maintaining consistency in an ontology should not be neglected. Ontologies are intended to store information, and creating a large ontology requires a great amount of work. Therefore, large ontologies will become valuable resources that are likely to stay in place over a longer period of time. Certainly, information in an ontology has to be updated, completed, revised and removed from time to time, a process which is often referred to as *ontology maintenance*:

> "Any ontology must be current and represent the 'here and now' to be properly understood, interpreted and acted upon. *Ontology maintenance* is the set of processes -– both manual and automatic -– that focus on keeping ontology representations current within the environment(s) where used." [Deg04]

However, maintaining consistency in a changing ontology is not only difficult, but also poses the question *if information should be removed at all*. By removing outdated facts, one loses the implicit information that these facts *were considered to be true* during a certain time span (i.e. while they were present in the ontology). For example, the concept "Soviet Union" is considered to be inexistent in 2005, but certainly existed from 1922–1991. By removing the concept from the ontology, the information that the Soviet Union existed at all would be lost. Contexts could help to deal with inconsistencies that arise from the maintenance process, e.g. by providing a certain context for all concepts and assertions that are considered to be outdated.

### 5.3.2 Possible Solutions

As already indicated, creating independent contexts for conflicting facts has shown to be a very promising solution to the problem of maintaining consistency. Such an approach has been taken by CYC, whose knowledge base contains a large number of so-called *microtheories*. Microtheories are constructs that represent different contexts and can be arranged in a polyhierarchy. A microtheory can be seen as a bundle of assertions that is based on a shared set of assumptions about a certain domain. Any microtheory is also implicitly based on the assumptions that are true in any of its parent microtheories in the hierarchy (the so-called "domain assumptions").
The most important aspect of a microtheory is that all assertions within a single context are consistent:

> "One of the functions of microtheories is to separate assertions into consistent bundles. *Within* a microtheory, the assertions must be mutually consistent. This means that no hard contradictions are allowed, and any apparent contradictions must be resolvable by evaluation of the evidence visible in that microtheory. In contrast, there may be inconsistencies *across* microtheories." [Fou]

A microtheory can also have a set of other microtheories that are consulted by the inference engine whenever a query can not be answered within the original context. All other facts are not available for reasoning purposes:

> "If there is a fact (or rule) somewhere in Cyc that's not in that Microtheory, and also not in any of the the other Microtheories accesible to that Microtheory [...], then Cyc will not use that fact" [Cyc02a]

Inaccessible facts are thus not to be processed by the inference engine.
Besides allowing for *global inconsistencies* while still allowing for *local inference*, microtheories are also helpful from a performance point of view. Since any query is to be answered by starting from a specific microtheory, the inference engine can focus on the context that is represented by that microtheory, which reduces search space and allows for more efficient inference mechanisms.
Summarized it can be said that representing contexts as distinct sets of assertions within one single ontology seems to be a solution that accounts for both global inconsistency (which is inevitable in a large scale ontology) and inference within the locally consistent context (which is necessary for deriving knowledge and answering queries). With respect to OWL, an implementation of a similar method is probably not straightforward, e.g. because of OWL's inability to represent generic if-then rules ("chains of properties", see Section 4.3.3). A proposal for introducing such rules to OWL is described in [MLYL04].

## 5.4 Dealing with Uncertainty

### 5.4.1 Problem Description

One more aspect of knowledge management has not been discussed so far, which is how *uncertain facts* are represented by ontologies. Human common sense includes much information that is based on believes and assumptions, and we commonly use words like "probably", "possibly" etc. to express uncertainty. Actually, a great part of human activities is based on assumptions with varying degree of certainty, mainly because it is usually too intricate to acquire all factual information before attempting further actions. Also, our everyday experience helps us to estimate the degree of uncertainty of a specific information; for instance, we assume that buildings, shops

etc. which we know to be located in a certain place today will probably also be there tomorrow, although this assumption may occasionally turn out to be wrong. On the other hand, we will not generally assume that e.g. vehicles will stay in place for a longer period of time.
The necessity of representing uncertain facts in knowledge bases is therefore well-known; again, it is somewhat surprising that neither OWL nor Topic Maps provide any support for capturing uncertainty. For OWL, this can be partially explained by its background in Description Logics:

> As with traditional crisp logic, any sentence in OWL, being asserted facts, domain knowledge, or reasoning results, must be either true or false and nothing in between. However, most real world domains contain uncertainty knowledge and incomplete or imprecise information that is true only to a certain degree. Ontologies defined by these languages thus cannot quantify the degree of the overlap or inclusion between two concepts, and cannot support reasoning in which only partial information about a concept or individual in the domain can be obtained." [DP04]

Dealing with uncertainty is especially important whenever inference is performed on an ontology. As Topic Maps are usually not directly related to inferencing, there is also no information about uncertainty within Topic Maps.

### 5.4.2 Possible Solutions

Traditionally, two different approaches can be taken to deal with uncertain facts in knowledgebases. The first approach is to abandon the monotonic nature of reasoning and is thus known as *non-monotonic reasoning*. The second approach is based on statistical considerations and commonly implemented with *Bayesian Networks*.
Within classical formal logics such as predicate logic, any new assertions that are introduced in a knowledge base must be mutually consistent with all other assertions that already exist in that knowledge base. It is impossible to create new assertions that retract any existing statements. Thus, these systems are said to be *monotonic*.
In order to deal with uncertain facts, monotonic reasoning is not appropriate since it does not allow for creating exceptions to otherwise generally valid rules. However, most human reasoning is exactly about creating generally valid rules and yet accepting exceptions to them. *Non-monotonic reasoning* tries to reproduce this kind of reasoning with approaches like *non-monotonic logic*, *default logic* and *circumscription*. The problem with non-monotonic reasoning is that although inference tasks are expected to become simpler and representation of concepts should be easier, the opposite is the case:

> "Although there are cases in which non-monotonic inference has a complexity which is comparable to classical inference, the general picture shows

> that tractable problems may become intractable [...], intractable problems may become 'more' intractable, decidable problems may become undecidable, and undecidable problems may become 'more' undecidable [...].
>
> [...]
>
> In fact, if we want to make inference more efficient, we have to give up either soundness or completeness. The general idea about NMR is that it can be seen as a fast but *unsound* approximation of ordinary reasoning." [CDS96]

Integration into OWL and possible extensions of the language are being investigated, but no results exist so far.
Statistical methods such as *Bayesian Network reasoning* have been used for OWL ontologies e.g. by Ding and Peng, in whose approach "the OWL language is first augmented to allow additional probabilistic markups so that probability values can be attached to individual concepts and properties as well as their interrelations in an OWL ontology" [DP04]. This is an example of the probabilistic markup syntax, taken from [DP04]:

```
<prob:PriorProbObj rdf:ID="P(Animal)">
  <prob:hasVariable>
    <rdf:value>&ont;Animal</rdf:value>
  </prob:hasVariable>
  <prob:hasProbValue>0.5</prob:hasProbValue>
</prob:PriorProbObj>
```

Results have been promising accoding to the amount of interest in the Semantic Web community.
Summarized, solutions for dealing with uncertainty are needed in order to represent knowledge that is closer to human common sense. While formal approaches such as non-monotonic reasoning have shown to be very complex, statistical methods may be an interesting alternative, as recent research activities show.

# 6 Conclusion

The significant interest in ontologies as "knowledge stores" is not surprising, as standards like OWL and Topic Maps offer a simple, yet powerful way to create ontologies and develop ontology-centric applications. The standards clearly address a very broad audience of researchers and developers and aim at the mainstream implementation of ontologies in applications that somehow deal with knowledge representation and knowledge management. The predefined ontological vocabularies are generic enough to facilitate the use of ontologies in almost any area of research.
Most problems that are related to ontologies are however not apparent at first sight, but emerge upon closer examination of the principles that form the foundations of ontologies. There are two distinguishable groups of issues related to ontologies, with different impacts on the future evolution and adoption of ontologies.
First, almost all currently available tools still focus on researchers in the area of ontology development as their primary target audience. For developers of standard software products or even end-users, practically no appropriate tools exist at the time of writing. This is not very surprising, as ontologies are still being researched mostly at a theoretic level with a lot of work yet to be done, e.g. in the field of inference engines. This observation can for instance also be made by looking at the query languages that are available today, which are often in experimental state and not yet intended for actual "every day use". Also, the process of exchanging information across ontology boundaries is still quite unclear and poorly defined, apart from the (obvious) use of URIs as global identifiers for concepts. Therefore, most projects that already use ontologies to store information usually concentrate only on their own, locally available ontologies, without placing much emphasis on information exchange. These group of issues is however not critical, as proposals for possible solutions exist that may be applied and improved in the near future.
The second group of problems is by far more crucial for the long-term success of ontologies, even though (or maybe because) they are not so apparent. Due to the existence of detailled standards and the large amount of work being done in the field of ontologies, certain very fundamental questions one might expect to be already answered are actually still heavily discussed, although they are very important even for the ontology standard itself. Many issues are even not discussed on a open basis, but appear to be deliberately ignored by certain parts of the community or dismissed with vague indications. This is for instance the case with the problematic notion of identity and the related questions of how to establish identity for arbitrary subjects in a

globally valid way (see Chapter 2). In my optinion, these questions are not adequately addressed by the RDF and OWL community, and also the Topic Map approach is far from being perfect, although establishing identity is one of the most important aspects of ontology design. Another example is the unrestricted use of URIs as only identifiers for all concepts, which is again addressed only in a very fuzzy way. Solution proposals often mistake *technical interoperability* (which is what URIs are actually intended for) for true *semantic interoperability* (which is independent from technical standards such as URIs).
These issues bring up the question whether ontologies can live up to the great expectations in context with the Semantic Web initiative: are ontologies suited for mainstream applications besides research projects and isolated solutions for some special applications? A strong contraindication is given by the nature of humans itself: as correctly pointed out by critics, humans tend to be lazy, imprecise, and not always commited to truth [Doc01]. It is therefore almost impossible to redefine human knowledge in such a precise, unambiguous way that is needed to represent that knowledge in an ontology. After all, it is an undeniable fact that the Semantic Web requires a transformation of knowledge representation from the natural language documents (books, web pages etc.) we use today to collection of ontologies that can also be interpreted by machines, although this pviewpoint is not very common among ontology researchers. Actually, this process is an adaption of human knowledge to a lower, "binary" level of machine interpretation, where only true or false statements exist. Humans can of course profit from this transformation process, because e.g. it allows for creating really precise search engines that reveal exactly those informations that we are looking for, but it is very questionable if such a transformation process can be successful at all, just by looking at the immense costs of such an undertaking.
There is however another, somewhat different way to take advantage of ontologies and their capabilities for improving knowledge management and automated processing of information. The basic idea is to combine natural language processing approaches (NLP) and ontologies in order to significantly improve the automated interpretation of ordinary documents originally intended for human interpretation. Ontologies could be used to represent some kind of common sense knowledge, which is indispensable for correct interpretation of information. By using an ontology, actual knowledge could be added to the syntactical and statistical methods used so far, allowing for semantic interpretation of a document. The key difference to the knowledge transformation process described above is that initially the ontology does not contain specific domain knowledge, but enables the interpreter to process documents and extract additional information from them. This will surely not solve all problems that are inherent to ontologies, but might facilitate the creation of large ontologies with domain specific knowledge. A number of works describing such systems exist [AKM$^+$03,HSC02,PSL03, NV04], but a in-depth discussion is outside the scope of this thesis.

# Bibliography


[ACKP01] Sofia Alexaki, Vassilis Christophides, Greg Karvounarakis, and Dimitris Plexousakis. On storing voluminous rdf descriptions: The case of web portal catalogs. In *Proceedings of the 4th International Workshop on the the Web and Databases*. ICS-FORTH, 2001. `http://athena.ics.forth.gr:9090/RDF/publications/webdb2001.pdf`.

[AdMRV02] Pascal Auillans, Patrice Ossona de Mendez, Pierre Rosenstiehl, and Bernard Vatant. A formal model for topic maps. In I. Horrocks and J. Hendler, editors, *LNCS*, volume 2342, pages 69–83. Springer-Verlag, 2002.

[Ahm00] Kal Ahmed. Topic maps for repositories. In *Proceedings of XML Europe 2000*, 2000. `http://www.gca.org/papers/xmleurope2000/pdf/s29-04.pdf`.

[Ahm03] Kal Ahmed. Topic map design patterns for information architecture. In *Proceedings of XML Conference & Exposition 2003*, 2003. `http://www.techquila.com/tmsinia.html`.

[Ahm04] Kal Ahmed. Topic maps – canonicalization, 2004. `http://www.isotopicmaps.org/sam/cxtm/`.

[AJ01] D. Albisser and A. Jetzer. Agent communication and semantic web, 2001. `http://www.jugs.ch/html/events/2001/SemanticWeb.pdf`.

[AKM+03] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel R. Shadbolt. Automatic ontology-based knowledge extraction from web documents. In *IEEE Intelligent Systems 18*, pages 14–21. IEEE, 2003.

[BA01] Olivier Baudon and Pascal Auillans. Graph clustering for very large topic maps. In *Proceedings of XML Europe 2001*, 2001. `http://www.gca.org/papers/xmleurope2001/papers/pdf/s23-2.pdf`.

[Bar03a] Robert Barta. Astma? language definition. Technical report, Bond University, 2003. `http://astma.it.bond.edu.au/astma%3F-spec.dbk`.

[Bar03b] Robert Barta. Astma? tutorial. Technical report, Bond University, 2003. `http://astma.it.bond.edu.au/astma%3F-tutorial.dbk`.

[Bar04] Robert Barta. Astma= language definition, 2004. `http://astma.it.bond.edu.au/astma=-spec-xtm.dbk`.

[BBN02] M. Biezunski, M. Bryan, and S. Newcomb. Iso/iec 13250, topic maps (second edition), 2002. `http://www.y12.doe.gov/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2%.pdf`.

[Beh03] Henning Behme. E-commerce für alle. *iX*, 7:21, 2003.

[BG02] Robert Barta and Jan Gylta. Xtmpath, manipulating topic map data structures, 2002. `http://topicmaps.it.bond.edu.au/docs/13?style=printable`.

[BGS+03] Henk Blanken, Torsten Grabs, Hans-Jörg Schek, Ralf Schenkel, and Gerhard Weikum, editors. *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*, volume 2818 of *LNCS*. Springer-Verlag, 2003.

[BGvH+03] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-owl: Contextualizing ontologies. In *Second International Semantic Web Conference ISWC'03*, volume 2870 of *LNCS*, page 164–179. Springer-Verlag, 2003. `http://www.cs.vu.nl/%7Eheiner/public/ISWC03.pdf`.

[BHS03] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In Dieter Hutter and Werner Stephan, editors, *Lecture Notes in Computer Science*, volume 2605. Springer-Verlag, 2003. `http://www.cs.man.ac.uk/~horrocks/Publications/download/2003/BaHS03.pdf%`.

[BK03] Jeen Broekstra and Arjohn Kampman. Inferencing and truth maintenance in rdf schema. In *Proceedings of PSSS*, 2003. `http://www.openrdf.org/doc/papers/inferencing.pdf`.

[BKvH01] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: An architecture for storing and querying rdf data and schema information. In H. Lieberman D. Fensel, J. Hendler and W. Wahlster, editors, *Semantics for the WWW*. MIT Press, 2001. `http://www.cs.vu.nl/~frankh/postscript/MIT01.pdf`.

[BKvH02] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In Ian Horrocks and James Hendler, editors, *Proceedings of International Semantic Web Conference ISWC 2002*, 2002. `http://www.openrdf.org/doc/papers/Sesame-ISWC2002.pdf`.

[BL00] Tim Berners-Lee. Semantic web on xml, 2000. `http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide1-0.html`.

[BLFIM98] T. Berners-Lee, R. Fielding, U.C. Irvine, and L. Masinter. Uniform resource identifiers (uri): Generic syntax. Technical report, IETF, 1998. `http://www.ietf.org/rfc/rfc2396.txt`.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, May 2001.

[BN01] Michael Biezunski and Steven R. Newcomb. Differences between xtm 1.0 and the hytime-based meta-dtd, 2001. `http://www.y12.doe.gov/sgml/sc34/document/0277.htm`.

[BNB02] M. Biezunski, S. Newcomb, and M. Bryan. Guide to the topic map standardization, 2002. `http://www.y12.doe.gov/sgml/sc34/document/0323.htm`.

[Bog04] Dmitry Bogachev. Tmpath – revisited, 2004. `http://homepage.mac.com/dmitryv/TopicMaps/TMPath/TMPathRevisited.html`.

[BOR] Bor – a pragmatic daml+oil reasoner. `http://www.ontotext.com/bor/`.

[Bou01a] Ronald Bourret. Mapping dtds to databases. *XML.com*, May 2001. `http://www.xml.com/lpt/a/2001/05/09/dtdtodbs.html`.

[Bou01b] Ronald Bourret. Xml-dbms, version 1.01, 2001. `http://www.rpbourret.com/xmldbms/readme.htm`.

[Bou03] Ronald Bourret. Xml and databases, 2003. `http://www.rpbourret.com/xml/XMLAndDatabases.htm`.

[Bra04] Terrence Brady. Representing software system information in a topic map. In *Proceedings of Extreme Markup Languages 2004*, 2004. `http://www.mulberrytech.com/Extreme/Proceedings/xslfo-pdf/2004/Brady01/%EML2004Brady01.pdf`.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein.

Owl web ontology language – reference, 2004. http://www.w3.org/TR/owl-ref/.

[Cas] The castor project. http://www.castor.org.

[CDS96] Marco Cadoli, Francesco M. Donini, and Marco Schaerf. Is intractability of non-monotonic reasoning a real drawback? *Artificial Intelligence*, 88:215–251, 1996. ftp://ftp.dis.uniroma1.it/pub/ai/papers/cado-doni-scha-95-b.ps.gz.

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. In *Proceedings of IJCAI 2003*, 2003. http://www.cs.vu.nl/~heiner/IJCAI-03/Papers/Chen.pdf.

[Cha01] Pierre-Antoine Champin. Rdf tutorial, 2001. http://www710.univ-lyon1.fr/~champin/rdf-tutorial/.

[CoB04] Context broker architecture – ontologies, 2004. http://cobra.umbc.edu/ontologies-2004-05.html.

[Com] Common logic standard. http://cl.tamu.edu.

[CvHH+01] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. Daml+oil (march 2001) reference description, 2001. http://www.w3.org/TR/daml+oil-reference.

[CVV04] David Celjuska and Maria Vargas-Vera. Semi-automatic population of ontologies from text. In Jan Paralic and Andreas Rauber, editors, *Proceedings of Workshop on Data Analysis WDA-2004*, 2004. http://kmi.open.ac.uk/publications/pdf/kmi-04-18.pdf.

[Cyc] Cycorp, inc. http://www.cyc.com.

[Cyc01] Draft version of the upper cyc ontology in xml topic map representation, 2001. http://xml.coverpages.org/ni2001-02-28-d.html.

[Cyc02a] Contexts in cyc®, 2002. http://www.cyc.com/cycdoc/course/contexts-basic-module.html.

[Cyc02b] Inc. Cycorp. The syntax of cycl. Technical report, Cycorp, Inc., 2002. http://www.cyc.com/cycdoc/ref/cycl-syntax.html.

[Deg04] Duane Degler. Maintaining ontology implementations: The value of listening. In *Proceedings of Extreme Markup Languages 2004*, 2004. http://www.mulberrytech.com/Extreme/Proceedings/xslfo-pdf/2004/Degler01%/EML2004Degler01.pdf.

[dG03] Marc de Graauw. Using topic maps to extend relational databases. *XML.com*, Mar 2003. http://www.xml.com/pub/a/2003/03/05/tmrdb.html.

[DMO] dmoz open directory project. http://www.dmoz.org.

[Doc01] Cory Doctorow. Metacrap: Putting the torch to seven straw-men of the meta-utopia, 2001. http://www.well.com/~doctorow/metacrap.htm.

[DOL04] Daml ontology library, 2004. http://www.daml.org/ontologies/.

[DP04] Zhongli Ding and Yun Peng. A probabilistic extension to ontology language owl., 2004. http://csdl.computer.org/comp/proceedings/hicss/2004/2056/04/205640111a.pdf.

[ES04] Marc Ehrig and York Sure. Ontology mapping – an integrated approach. In *Proceedings of the 1st European Semantic Web Symposium ESWS2004*, 2004. http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/2004_esws_mapping.pdf.

[eXi] exist – open source native xml database. http://exist.sourceforge.net/.

[FDES98] Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer. Ontobroker: Or how to enable intelligent access to the www. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management (KAW '98), Banff, Canada*, APR 1998. http://www.ubka.uni-karlsruhe.de/cgi-bin/psgunzip/1998/wiwi/9/9.pdf.

[Fit02] Kent Fitch. Taking rdf and topic maps seriously. In *Proceedings of AUSWEB02*, 2002. http://ausweb.scu.edu.au/aw02/papers/refereed/fitch2/paper.html.

[Fou] Foundations of knowledge representation in cyc. http://www.cyc.com/doc/tut/DnLoad/Microtheories.pdf.

[Gar02] Lars Marius Garshol. The linear topic map notation. Technical report, Ontopia, 2002. http://www.ontopia.net/download/ltm.html.

[Gar03a] Garshol. Living with topic maps and rdf. Technical report, Ontopia, 2003. http://www.ontopia.net/topicmaps/materials/tmrdf.html.

[Gar03b] Lars Marius Garshol. tolog 1.0 – ontopia technical report 11 12 2003. Technical report, Ontopia AS, 2003. http://www.ontopia.net/topicmaps/materials/tolog-spec.html; incomplete draft at the time of writing.

[Gar03c] James Garson. Stanford encyclopedia of philosophy, 2003. `http://plato.stanford.edu/`.

[Gar04a] Lars Marius Garshol. Metadata? thesauri? taxonomies? topic maps! Technical report, Ontopia, 2004. `http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html`.

[Gar04b] Lars Marius Garshol. tolog language tutorial. Technical report, Ontopia, 2004. `http://www.ontopia.net/omnigator/docs/query/tutorial.html`.

[GB03a] Lars Marius Garshol and Robert Barta. Tmql requirements, 2003. `http://www.isotopicmaps.org/tmql/tmqlreqs.html`.

[GB03b] Lars Marius Garshol and Robert Barta. Topic map query language, use cases, 2003. `http://www.isotopicmaps.org/tmql/use-cases.html`.

[GLR] Glossary of linguistics and rhetoric. `http://rinkworks.com/words/linguistics.shtml`.

[GM05] Lars Marius Garshol and Graham Moore. Topic maps – data model, 2005. `http://www.isotopicmaps.org/sam/sam-model/`.

[Gra01] Mark Graves. *Designing XML Databases*. Prentice Hall, October 2001.

[Gro01] TopicMaps.org Authoring Group. Xml topic maps (xtm) 1.0, topicmaps.org specification, 2001. `http://www.topicmaps.org/xtm/1.0/`.

[Gru95] Tom R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995. `ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-93-04.ps.gz`.

[Guh] R. V. Guha. rdfdb: An rdf database. `http://www.guha.com/rdfdb/`.

[Haw03] Sandro Hawke. Disambiguating rdf identifiers, 2003. `http://www.w3.org/2002/12/rdf-identifiers/`.

[Hib] Hibernate. `http://www.hibernate.org`.

[HM01] Volker Haarslev and Ralf Möller. Racer system description. In *Proceedings of International Joint Conference on Automated Reasoning (IJCAR'2001)*, volume 2083 of *LNAI*, pages 701–705. Springer-Verlag, 2001. `http://www.cs.concordia.ca/~haarslev/publications/ijcar-2001-racer.pdf`.

[Hor01] Ian Horrocks. The fact system, 2001. http://www.cs.man.ac.uk/~horrocks/FaCT/.

[HPSvH03] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1), 2003.

[HS01] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D})$ description logic. In *Proceedings of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204, 2001. http://www.cs.man.ac.uk/~horrocks/Publications/download/2001/ijcai01.pd%f.

[HSC02] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-cream – semi-automatic creation of metadata. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*, pages 358–372, 2002. http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/ekaw2002scr%eam-sub.pdf.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th Int. Conf. on Logic Programming and Automated Reasoning (LPAR'99)*, number 1705 in LNAI, pages 161–180. Springer-Verlag, 1999. http://www.cs.man.ac.uk/~horrocks/Publications/download/1999/lpar99.pdf%.

[HT02] Ian Horrocks and Sergio Tessaris. Querying the semantic web: A formal approach. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 177–191, 2002. http://www.cs.man.ac.uk/~horrocks/Publications/download/2002/iswc2002.pdf.

[IM04] Thomas M. Insalaco and James David Mason. Navigating the production maze: The topic mapped enterprise. In *Proceedings of Extreme Markup Languages 2004*, 2004. http://www.mulberrytech.com/Extreme/Proceedings/xslfo-pdf/2004/Mason01/%EML2004Mason01.pdf.

[Jen] Jena -– a semantic web framework for java. http://jena.sourceforge.net/.

[Jen04] Jena2 database interface – database layout, 2004. http://jena.sourceforge.net/DB/layout.html.

[KIF] Suo-kif. http://suo.ieee.org/SUO/KIF/suo-kif.html (draft proposal without official status).

[KK03] Alexander Kuckelberg and Ralph Krieger. Efficient structure oriented storage of xml documents using ordbms. In *LNCS*, volume 2590, pages 131–141. Springer-Verlag, 2003.

[KLS+01] Josef Küng, Thomas Luckeneder, Knud Steiner, Roland Wagner, and Wolfram Wöß. Persistent topic maps for knowledge and web content management. In *Proceedings of WISE (2)*, pages 151–158, 2001.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of ACM*, 42:741–843, July 1995. `http://citeseer.ist.psu.edu/kifer90logical.html`.

[KMA+02] Gregory Karvounarakis, Aimilia Magkanaraki, Sophia Alexaki, Vassilis Christophides, Dimitris Plexousakis, Michel Scholl, and Karsten Tolle. Rql: A functional query language for rdf. In *Proceedings of The Eleventh International World Wide Web Conference WWW2002*, 2002. `http://athena.ics.forth.gr:9090/RDF/publications/FuncBook.pdf`.

[KSD01] Atanas K. Kiyakov, Kiril Iv. Simov, and Marin Dimitrov. Ontomap: Ontologies for lexical semantics. In *Proceedings of Semantic Web Working Symposium (SWWS)*, 2001. `http://www.ontotext.com/publications/ranlp01.pdf`.

[Ksi00] Rafal Ksiezyk. Answer is just a question [of matching topic maps]. In *Proceedings of XML Europe 2000*, 2000. `http://www.gca.org/papers/xmleurope2000/pdf/s22-03.pdf`.

[Kuh99] Markus Kuhn. Utf-8 and unicode faq for unix/linux, 1999. `http://www.cl.cam.ac.uk/~mgk25/unicode.html`.

[Las02] Ora Lassila. Taking the rdf model theory out for a spin. In *The Semantic Web – ISWC 2002*, volume 2342 of *LNCS*, pages 307–317. Springer-Verlag, 2002. `http://www.lassila.org/publications/2002/lassila-iswc2002.pdf`.

[Lee04] Ryan Lee. Scalability report on triple store applications. Technical report, MIT, 2004. `http://simile.mit.edu/reports/stores/stores.pdf`.

[Lis] Stefan Lischke. Xtm4xmldb. `http://sourceforge.net/projects/xtm4xmldb`.

[Mit03] Mittermeier. Naiv nativ. *iX*, 8:42, 2003.

[MLYL04] Jing Mei, Shengping Liu, Anbu Yue, and Zuoquan Lin. An extension to owl with general rules. In *Proceedings of RuleML*, volume 3323 of *Lecture*

*Notes in Computer Science*. Springer, 2004. http://www.is.pku.edu.cn/~lz/publications/owlrule.pdf.

[MN03] Graham Moore and Mary Nishikawa. Topic map constraint language (straw man proposal), 2003. http://www.isotopicmaps.org/tmcl/tmcl.html.

[MNJ04] Prasenjit Mitra, Natalya F. Noy, and Anuj R. Jaiswal. Omen: A probabilistic ontology mapping tool. In *Proceedings of ISWC-04 Workshop on Meaning Coordination and Negotiation held in conjunction with 3rd International Semantic Web Conference (ISWC-2004)*, 2004. http://smi.stanford.edu/people/noy/papers/omen_ISWC.pdf.

[MNV02] Michele Missikoff, Roberto Navigli, and Paola Velardi. The usable ontology: An environment for building and assessing a domain ontology. In I. Horrocks and J. Hendler, editors, *Proceedings of ISWC 2002*, volume 2342 of *LNCS*, pages 39–53. Springer-Verlag, 2002.

[NB00] Steven Newcomb and Michael Biezunski. Topic maps go xml. In *Proceedings of XML Europe 2000*, 2000. http://www.gca.org/papers/xmleurope2000/pdf/s11-02.pdf.

[New02] Steven R. Newcomb. Preemptive reification. In *Lecture Notes On Computer Science*, volume 2342, pages 414–418. Springer-Verlag, 2002.

[NP01] Ian Niles and Adam Pease. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001. http://projects.teknowledge.com/HPKB/Publications/FOIS.pdf.

[NV04] Roberto Navigli and Paola Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30(2), 2004. http://www.dsi.uniroma1.it/~velardi/CL.pdf.

[NZE05] New zealand electronic text centre – topic maps, 2005. http://www.nzetc.org/tm/scholarly/tei-NZETC-About-technology-topicmap.h%tml.

[OFT04] Owl web ontology langugae – overview, 2004. http://www.w3.org/TR/owl-features/.

[Ogb00] Uche Ogbuji. An introduction to rdf. Technical report, IBM, 2000. ftp://www6.software.ibm.com/software/developer/library/w-rdf.pdf.

[OGD04] Owl web ontology language – guide, 2004. `http://www.w3.org/TR/owl-guide/`.

[OLB] A library of owl ontologies. `http://protege.stanford.edu/plugins/owl/owl-library/index.html`.

[OO02] Roxane Ouellet and Uche Ogbuji. Introduction to daml. *XML.com*, Jan 2002. `http://www.xml.com/pub/a/2002/01/30/daml1.html`.

[OSM04] Owl web ontology language – semantics and abstract syntax, 2004. `http://www.w3.org/TR/owl-semantics/`.

[OUC04] Owl web ontology language – use cases and requirements, 2004. `http://www.w3.org/TR/webont-req/`.

[PBZ04] L. Serafini P. Bouquet and S. Zanobini. Peer-to-peer semantic coordination. In *Proceedings of Semantic Web Applications and Perspectives (SWAP) 2004*, 2004. `http://semanticweb.deit.univpm.it/swap2004/cameraready/bouquet.pdf`.

[Pep02a] Steve Pepper. Lessons on applying topic maps. Technical report, Ontopia, 2002. `http://www.ontopia.net/topicmaps/materials/xmlconf.html`.

[Pep02b] Steve Pepper. Ten theses on topic maps and rdf. Technical report, Ontopia, 2002. `http://www.ontopia.net/topicmaps/materials/rdf.html`.

[Pep03] Steve Pepper. Published subjects: Introduction and basic requirements. Technical report, OASIS, 2003. `http://www.ontopia.net/tmp/pubsubj-gentle-intro.htm`.

[PH04] Feng Pan and Jerry R. Hobbs. Time in owl-s. In *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, 2004. `http://www.isi.edu/~pan/damltime/AAAIsymp2004.pdf`.

[Pin04] Rani Pinchuk. Toma – topic map query language. Technical report, Space Applications Services, 2004. `http://www.spaceapplications.com/toma/Toma.html`.

[PS03] Steve Pepper and Sylvia Schwab. Curing the web's identity crisis. Technical report, Ontopia, 2003. `http://www.ontopia.net/topicmaps/materials/identitycrisis.html`.

[PSL03] Chintan Patel, Kaustubh Supekar, and Yugyung Lee. Ontogenie: Extracting ontology instances from www. In *Proceedings of ISWC2003*, 2003. http://www.sice.umkc.edu/~leeyu/Publications/spaper1.pdf.

[QG04] Yuzhong Qu and Zhiqiang Gao. Interpreting distributed ontologies, 2004. http://www.www2004.org/proceedings/docs/2p270.pdf.

[RAC] Racer system description. http://www.sts.tu-harburg.de/~r.f.moeller/racer/.

[Rat99] Hans Holger Rath. Mozart und Kugeln. *iX*, 12:149, 1999.

[Rat03] Hans Holger Rath. The topic maps handbook. Technical report, Empolis, 2003. http://www.empolis.com/downloads/empolis_TopicMaps_Whitepaper20030206.p%df.

[RL02] Stephen L. Reed and Douglas B. Lenat. Mapping ontologies into cyc. Technical report, Cycorp, Inc., 2002. http://www.cyc.com/doc/white_papers/mapping-ontologies-into-cyc_v31.pdf%.

[RML] The rule markup initiative. http://www.ruleml.org/.

[RPR04] Rdf primer, 2004. http://www.w3.org/TR/rdf-primer/.

[RTS] 3store RDF triple store. http://sourceforge.net/projects/threestore/.

[SCPV04] Stefano Spaccapietra, Nadine Cullot, Christine Parent, and Christelle Vangenot. On spatial ontologies. In *Proceedings of GEOINFO2004*, 2004. http://lbdwww.epfl.ch/~stefano/geoinfolast.pdf.

[SD02] Michael Sintek and Stefan Decker. Triple – a query, inference, and transformation language for the semantic web. In *Proceedings of ISWC 2002*, 2002. http://triple.semanticweb.org/doc/iswc2002/TripleReport.pdf.

[Sea04a] Andy Seaborne. Rdql – a query language for rdf. Technical report, HP Labs Bristol, 2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.

[Sea04b] Andy Seaborne. Sparql query language for rdf. Technical report, HP Labs Bristol, 2004. http://www.w3.org/TR/rdf-sparql-query/.

[SeR] User guide for sesame – the serql query language, rev. 1.1. http://www.openrdf.org/doc/users/ch06.html.

[SEW01] Knud Steiner, Wolfgang Essmayr, and Roland Wagner. Topic maps – an enabling technology for knowledge management. In *Proceedings of 2nd Int. Workshop on Theory and Applications of Knowledge Management (TAKMA'01), in conjunction with DEXA'01*, 2001.

[SGKM] Nick Siegel, Keith Goolsbey, Robert Kahlert, and Gavin Matthews. The cyc system: Notes on architecture. `http://www.cyc.com/cyc/technology/whitepapers_dir/Cyc_Architecture_and_%API.pdf`.

[Sow] John Sowa. A brief introduction to conceptual graphs. `http://www.cs.uah.edu/~delugach/CG/Sowa-intro.html`.

[Sto98] David G. Stork, editor. *HAL's Legacy: 2001's Computer as Dream and Reality*. MIT Press, 1998.

[SUM] Sumo, suggested upper merged ontology. `http://topicmaps.bond.edu.au/mda/internet/semantic-web/ontology/sumo`.

[SUO] Ieee p1600.1 – standard upper ontology working group (suo wg) – scope & purpose. `http://suo.ieee.org/SUO/scopeAndPurpose.html`.

[Tau02] Carsten Tautz. The role of ontologies and taxonomies in knowledge technologies. Technical report, Empolis, 2002. `http://www.knowledgetechnologies.net/proceedings/presentations/tautz/kt%2002-tautz-role-of-ontologies.pdf`.

[TMQ] Topic map query language (tmql). `http://www.isotopicmaps.org/tmql/`.

[UC] Unicode consortium. `http://www.unicode.org/`.

[Var02] Vasudeva Varma. Building large scale ontology networks. Technical report, International Institute of Information Technology Hyderabad, 2002.

[Vat03] Bernard Vatant. Owl and topic map pudding, 2003. `http://www.mondeca.com/owl/owltm.htm`.

[Vat04] Bernard Vatant. Ontology-driven topic maps. In *Proceedings of XML Europe 2004*, 2004. `http://www.idealliance.org/europe/04/call/xmlpapers/03-03-03.91/.03-03-%03.html`.

[VDL04] Rdf vocabulary description language 1.0: Rdf schema, 2004. `http://www.w3.org/TR/rdf-schema/`.

[WKLW98] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin core metadata for resource discovery. Technical report, IETF, 1998. `http://www.ietf.org/rfc/rfc2413.txt`.

[WM01] Richard Widhalm and Thomas Mück. *Topic Maps: Semantische Suche im Internet*. Springer-Verlag, 2001.

[Wor] Wordnet – a lexical database for the english language. `http://www.cogsci.princeton.edu/~wn/`.

[XQR04] Xquery, 2004. `http://www.w3.org/TR/xquery/`, `http://www.w3.org/TR/xquery-semantics/`.

[XSB] Xsb logic programming and deductive database system. `http://xsb.sourceforge.net/`.

[YKZ03] Guizhen Yang, Michael Kifer, and Chang Zhao. Flora-2: A rule-based knowledge representation and inference infrastructure for the semantic web. In *Proceedings of Second International Conference on Ontologies, Databases and Applications of Semantics (ODBASE)*, 2003. `http://www.cse.buffalo.edu/faculty/gzyang/papers/odbase2003.pdf`.

[ZFC04] Youyong Zou, Timothy W. Finin, and Harry Chen. F-OWL: An inference engine for semantic web. In *Proceedings of FAABS*, pages 238–248, 2004.