

# Logic-Based Open Hypermedia for the Semantic Web

Peter Dolog<sup>1</sup>, Nicola Henze<sup>2</sup>, and Wolfgang Nejdl<sup>1,2</sup>

<sup>1</sup> Learning Lab Lower Saxony,  
University of Hannover,  
Expo Plaza 1, D-30539 Hannover, Germany

{dolog,nejdl}@learninglab.de  
<http://www.learninglab.de/~dolog>

<sup>2</sup> ISI- Knowledge-Based Systems,  
University of Hannover,  
Appelstr. 4, D-30167 Hannover, Germany

henze@kbs.uni-hannover.de  
<http://www.kbs.uni-hannover.de/~{henze,nejdl}>

**Abstract.** A challenge of the semantic web is provision of distributed information with well defined meaning, understandable for different parties. The aim of open hypermedia research has been to support such understanding on hypertext structures since several years. In this paper we show how the semantic web resource description formats can be utilized for automatic generation of hypertext structures from distributed metadata. Ontologies and metadata for three types of resources (domain, user, and observation) are investigated. We investigate a logic-based approach to open hypermedia using TRIPLE, rule-based language for the semantic web.

**keywords:** open hypermedia, semantic web, ontologies, adaptive hypermedia.

## 1 Introduction

The vision of the semantic web is to enable machines to interpret and process Information in the World Wide Web. The aim is to support humans in carrying out their various tasks with the World Wide Web. Several technologies have been developed for shaping, constructing and developing the semantic web. Many of the so far developed semantic web technologies provide us with tools for describing and annotating resources on the Web in standardized ways, e.g. with the Resource Description Framework (RDF [25]) and its binding to XML (eXtensible Markup Language [28]).

In this paper we will show how semantic web technologies can be used for building interoperable open hypermedia systems. In particular, we show how rules can be enabled to reason over distributed information resources in order to dynamically derive hypertext relations. On the Web, information can be found in various resources (e.g. documents), in annotation of these resources (like RDF-annotations on the documents themselves), in metadata files (like RDF descriptions), or in ontologies. Based on these sources of information we can think of functionality allowing us to derive new relations between information.

To describe and implement this functionality, there are at least two related research areas which contribute: *open hypermedia* and *adaptive hypermedia*. Techniques developed in open hypermedia have especially investigated how to (dynamically) construct hypermedia systems based on ontologies for document metadata. From the area of adaptive hypermedia, methods and techniques have been developed to adapt a hypermedia system to the various needs of the individual user or groups of users. We aim to join these different research areas in order to provide reasoning functionality for open, distributed information environments based on semantic web technologies. The result is an environment for reasoning on the semantic web, powered by ontologies and RDF-descriptions, where reasoning rules derive personalized hypertext relations.

The paper is structured as follows: section 2 describes the representation of resources with semantic web technologies, and shows our use of a domain, user, and observation ontologies. The following section (section 3) discusses our approach to generate hypertext structures / associations, and an example set of rules for dynamically generating personalized associations between information. A comparison of our approach to related work and a conclusion end the paper.

## 2 Representation of Resources

Semantic web technologies like the Resource Description Format (RDF) [20] or RDF schema (RDFS) [25] provide us with interesting possibilities. RDF schemas serve to define vocabularies for metadata records in an RDF file. RDF schemas can be used to describe resources, e.g. the RDF bindings of Learning Object Metadata (LOM) [23] can be used for these purposes, or RDF bindings of Dublin Core [13]. There is no restriction on the use of different schemas together in one RDF file or RDF model. The schema identification comes with attributes being used from that schema so backward dereferencing is again easily possible.

While RDF schema provides a simple ontology language, more powerful ontology languages which reside on top of RDF and RDF schema are available, too. For example, ontology languages like DAML+OIL [8] (the joint initiative of DAML (Darpa Agent Markup Language) and OIL (Ontology Inference Layer)) provide ontology layers on top of RDF / XML. Recently, OWL [24] (Web Ontology Language) has been developed, further enriching RDF.

An open question is how we can combine reasoning mechanisms on these (distributed) metadata and data resources, in order to generate hypertext presentations, link structures, etc., to bring the interoperability ideas from OHS to the WWW. This section will first describe semantic web tools that we employ in our approach, and then describe some structures for metadata components which allow us to generate link structures according to user features.

### 2.1 Bringing together Resources and Reasoning

On top of the RDF and ontology-layer, we find the layer of logic in the semantic web tower, or, more recently, the layers of rules and logic framework [3]. In our approach, the communication between reasoning rules and the open information environment will take place by exchanging RDF annotations: the rules reason over distributed RDF-annotations, results will be given back as RDF-files, too.

A rule language especially designed for querying and transforming RDF models is TRIPLE [26]. Rules defined in TRIPLE can reason about RDF-annotated information resources (required translation tools from RDF to triple and vice versa are provided).

TRIPLE supports *namespaces* by declaring them in clause-like constructs of the form *namespace-abbrev := namespace*, resources can use these namespaces abbreviations. *Statements* are similar to F-Logic object syntax: An RDF statement (which is a triple) is written as *subject[predicate → object]*. Several statements with the same subject can be abbreviated in the following way:

```
sun_java: 'index.html' [rdf:type->doc:Document;  
  doc:hasDocumentType->doc:StudyMaterial].
```

RDF *models* are explicitly available in TRIPLE: Statements that are true in a specific model are written as "@model", e.g.

```
doc:00_Class[rdf:type->doc:Concept]@results:simple.
```

Connectives and quantifiers for building logical formulae from statements are allowed as usual, i.e.  $\wedge$ ,  $\vee$ ,  $\neg$ ,  $\forall$ ,  $\exists$ , etc. For TRIPLE programs in plain ASCII syntax, the symbols AND, OR, NOT, FORALL, EXISTS, <- , ->, etc. are used. All variables must be introduced via quantifiers, therefore marking them is not necessary.

### 2.2 Domain Ontologies

First of all we need to determine a domain models (ontologies). The domain models comprise usually classes (classifies objects from a domain) and relationships between them. One possible domain in hypermedia application can be a domain of documents and concepts described in an application domain.

A simple ontology for documents and their relationships to other components is depicted in fig. 1. The class **Document** is used to annotate a resource which is a document. Documents describe some concepts. We use class **Concept** to annotate concepts. Concepts and documents are related through

dc:subject property. Documents can be ordered by dcterms:requires relationship. Concepts and documents have a certain role in their collaboration in certain document. We represent these facts by instances of DocumentRole class and its two properties: isPlayedIn and isPlayedBy. Concepts, document roles and concept roles can form hierarchies. We define subRoleOf, subConceptRoleOf, and subConceptOf properties for these purposes.

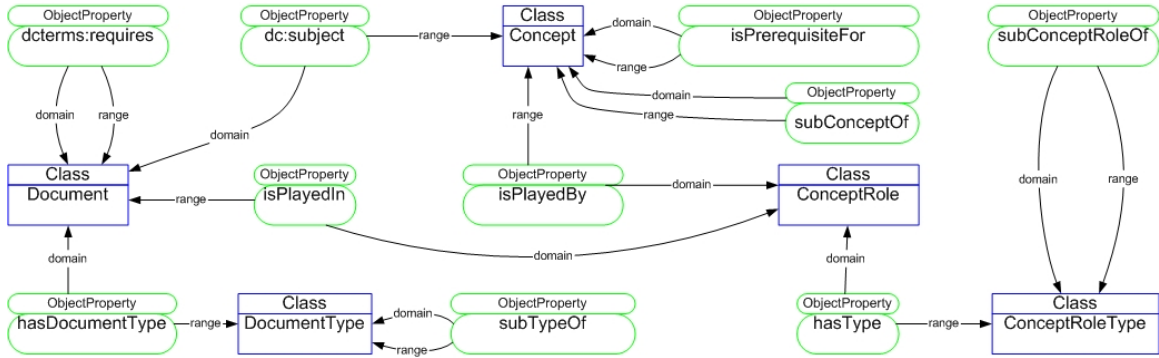


Fig. 1. Ontology of documents

Concepts play a certain role in a document. We recognize `Introduction` and `FullDescription` concept roles.

Document can have a type. One of application area where we can see the use of different document types is education. Figure 2 depicts the ontology with several document types for educational domain. The model is sometimes called `representation domain model` [9]. The most general document type is `Educational Material`. `Educational Material` has two subtypes: `Course Material` and `Examination Material`. `Examination Material` can be further specialized to `Project Task`, `Exam Task`, and `Exam`. The `Exam` can consist of the `Exam Task`-s.

`Course Material` can be further specialized into `Lecture`, `Example`, `LectureNote`, `Course`, `Exercise`, and `Project Assignment`.

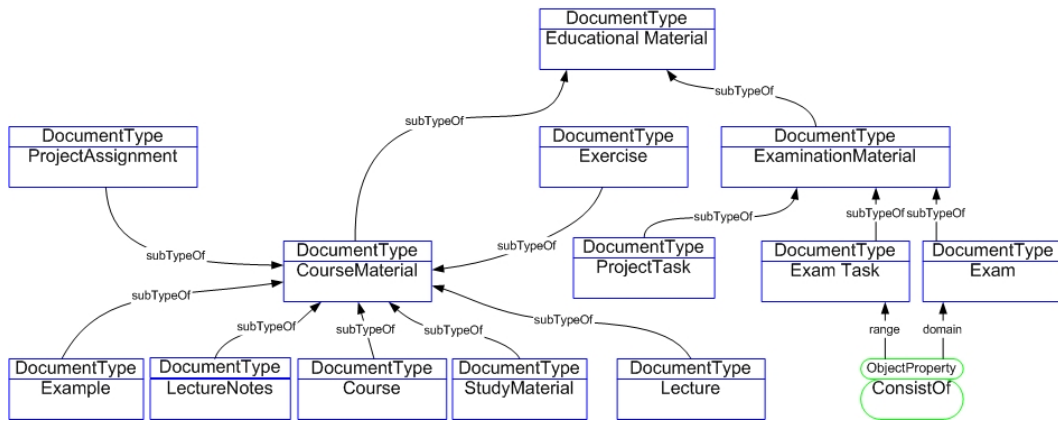
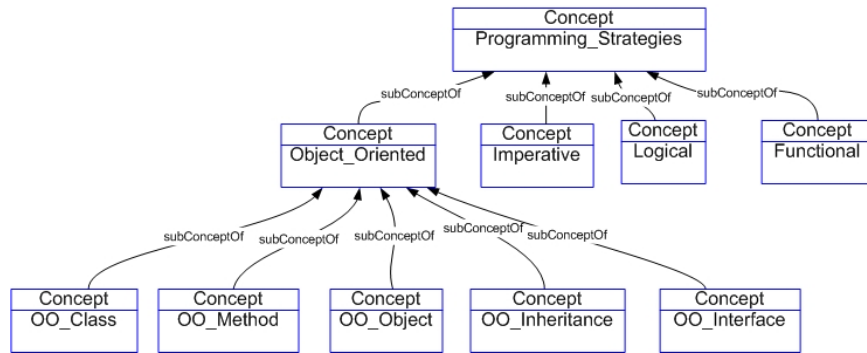


Fig. 2. Ontology for documents types

Figure 3 depicts `ProgrammingStrategies` concept with its subconcepts: `ObjectOriented`, `Imperative`, `Logical`, and `Functional`. `OOClass`, `OOMethod`, `OOObject`, `OOInheritance`, and `OOInterface` are depicted as subconcepts of `ObjectOriented`. The model is sometimes called `application domain model` [9].



**Fig. 3.** Concept ontology for Java e-lecture

Above described ontologies are used then in annotations of concrete documents/resources. An example of such resource can be a page describing `sun_java:'java/concepts/class.html'`. Following example shows how such a page can be annotated based on ontologies.

```
sun_java:'java/concepts/class.html' [
rdf:type->doc:Document;
dc:subject->doc:OO_Class].

doc:OO_Class[
rdf:type->doc:Concept;
doc:isPrerequisiteFor->doc:OO_Inheritance;
doc:subConceptOf->doc:Classes_and_objects].

doc:ClassesIntroduction[
rdf:type->doc:ConceptRole;
doc:isPlayedBy->doc:OO_Class;
doc:isPlayedIn->sun_java:'java/concepts/class.html';
doc:hasType->doc:Introduction].

doc:Introduction[
rdf:Type->doc:ConceptRoleType;
doc:subConceptRoleOf->doc:Cover].
```

The page is a document (RDF type `Document`). It describes information about classes. Thus it is annotated with `OO_Class` concept covered in the page. The `OO_Class` concept is annotated with type `Concept` and is subconcept of the `Classes_and_objects` concept. The `OO_Class` concept is prerequisite for the `OO_Inheritance`. A page can have prerequisites. Then the `dcterms:requires` property can be used in the annotation.

The `OO_Class` concept plays a role of introduction in the `sun_java:'java/concepts/class.html'` document. This is annotated by `ClassesIntroduction` resource, which is of type `ConceptRole`. The reference to `OO_Class` concept and the document where it plays the introduction role is annotated by using properties `isPlayedBy` and `isPlayedIn` respectively. The role has type `Introduction`. The `Introduction` is of type `ConceptRoleType` and is subtype of `Cover` concept role type.

### 2.3 Users

Data about a user serves for deriving contextual structures. It is used to determine how to adapt presentation of hypertext structures. Here we define an ontology for a user profile based on IEEE Personal and Private Information (PAPI) [18]. PAPI distinguishes *personal*, *relations*, *security*, *preference*, *performance*, and *portfolio* information. The *personal* category contains information about names, contacts and addresses of a user. *Relations* category serves as a category for specifying relationships between users (e.g. *classmate*, *teacherIs*, *teacherOf*, *instructorIs*, *instructorOf*, *belongsTo*, *belongsWith*). *Security* aims to provide slots for credentials and access rights. *Preference* indicates

the types of devices and objects, which the user is able to recognize. *Performance* is for storing information about measured performance of a user through learning material (i.e. what does a user know). *Portfolio* is for accessing previous experience of a user. Each category can be extended. For more discussion on learner modeling standards see for example [12].

Figure 4 depicts an example of an ontology for a learner profile. The ontology is based on *performance* category of PAPI. We are storing sentences about a learner which has a **Performance**. The **Performance** is based on learning experience (**learningExperienceIdentifier**), which is taken from particular document. The experience implies a **Concept** learned from the experience, which is maintained by **learningCompetency** property. The **Performance** is certified by a **Certificate**, which is issued by a certain **Institution**. The **Performance** has a certain **PerformanceValue**, which is in this context defined as a float number and restricted to interval from 0 to 1.

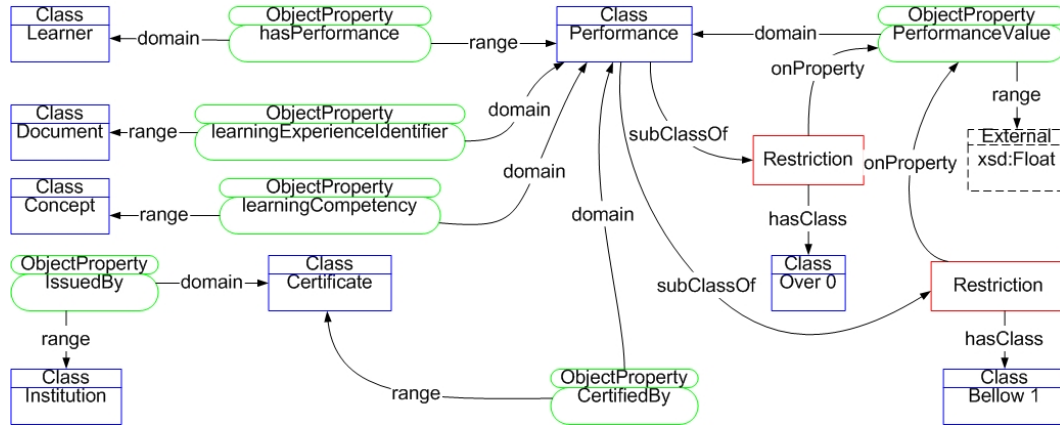


Fig. 4. Ontology for learner performance

The example of simple learner profile can look as follows.

```

user:user2[
  rdf:type -> learner:Learner;
  learner:hasPerformance -> user:user2P].

user:user2P[
  rdf:type->learner:Performance;
  learner:learningExperienceIdentifier->sun_java:'java/concepts/object.html';
  learner:learningCompetency->doc:00_Object;
  learner:CertifiedBy->KBScerturi:C1X5TZ3;
  learner:PerformanceValue->0.9
].

KBScerturi:C1X5TZ3[
  rdf:type->learner:Certificate;
  learner:IssuedBy->KBSuri:KBS
].

KBSuri:KBS[
  rdf:type->learner:Institution
].

```

The learner **user2** has the performance (**user2P**) record. The performance contains a learning experience about the KBS Java objects resource. The concept covered in the resource is stored in the performance as well. Then a certificate about the performance with performance value and institution who issued the certificate is recorded into the learner performance as well.

## 2.4 Observations

During runtime, users interact with a hypertext system. The user's interactions can be used to draw conclusions about possible user interests, about user's goal, user's task, user's knowledge, etc. These concluded user features can, as described in the previous section, be used for providing personalized views on hypertexts. An ontology of observations should therefore provide a structure of information about possible user observations, and - if applicable - their relations and/or dependencies.

A simple ontology for observations is depicted in fig. 5. The ontology allow us to instantiate facts that a **Learner** has interacted with (**hasInteraction** property) with a particular **Document** (**isAbout** property) via an interaction of a specific type (**InteractionType**). The interaction has taken place in a time interval between **beginTime** and **endTime**, and has a certain level (**Level**) associated, the **ObservationLevel**. Several events (see next section) can contribute to an interaction. Example of **InteractionTypes** are of kind **access**, **bookmark**, **annotate**, examples for **ObservationLevels** are that a user has **visited** a page, has **worked** on a project, has **solved** some exercise, etc.

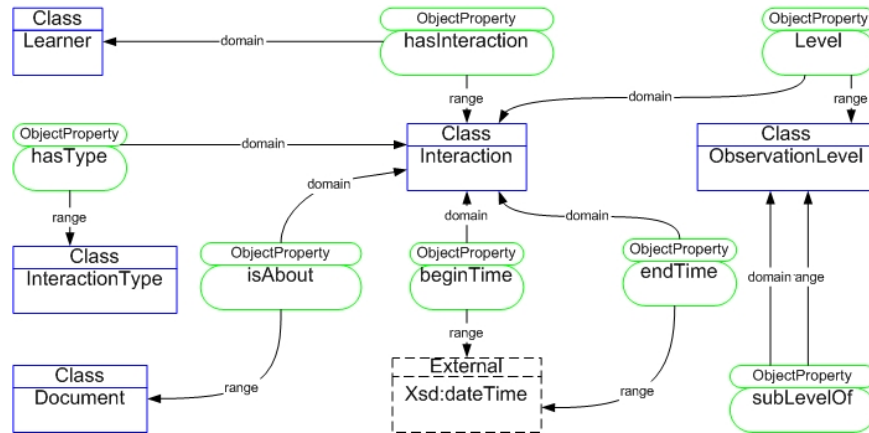


Fig. 5. Ontology for observations

## 3 Generating Hypertext Structures

Hypertext structures as described in several works on open hypermedia (see e.g [21]) can be generated from metadata reported in the previous section. We do not store the hypertext structures on servers as first class entities but we allow to generate such structures on the fly. In order to generate such hypertext structures we need an ontology for structures. Then transformation rules can be used to generate instances of that structure.

### 3.1 Presentation Ontology

A presentation ontology is used for describing structure relevant for visualization. Such an ontology adapted from FOHM [21] is depicted in fig. 6.

The main element of the ontology is the **Association**. Like in [21], the **Association** is built from three components: **Bindings**, **RelationType**, and **StructuralType** (in FOHM they refer to it as Cartesian product of bindings, relation type and structural type). These three components (classes) are related to association through **hasBindings**, **hasRelationType**, and **hasStructuralType** properties.

**Bindings** references a particular **Resource** on the web (document, another association, etc.), and **Feature**-s. A **Feature** can be a **Direction**, **Shape**, etc. Entries for **Direction** are depicted in figure 7b, entries for **Shape** are depicted in the figure 7c.

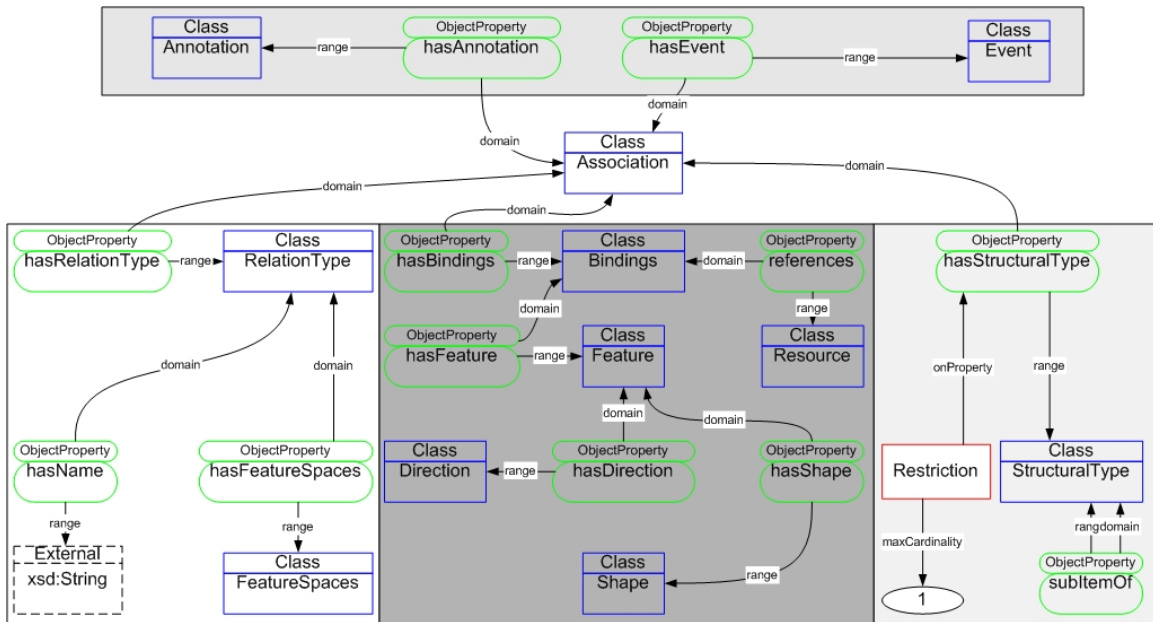


Fig. 6. A part of presentation ontology

The `RelationType` has a `Name` which is a string. The `RelationType` also points to the `FeatureSpaces`. Entries for the `FeatureSpaces` are depicted in figure 7a. A `StructuralType` is one of stack, link, bag, or sequence of resources.

In addition, `Association` can have associated events (e.g. click events for processing user interactions) through `hasEvent` property, and an annotation (e.g. green/red/yellow icon from traffic light metaphor technique from adaptive hypermedia [4]) through `hasAnnotation` property.

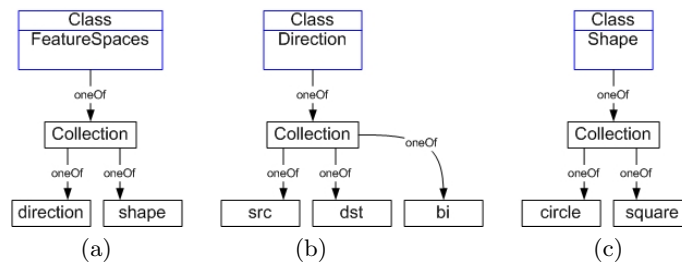


Fig. 7. Members of Collection of: (a) Feature Spaces, (b) Direction, (c) Shape.

The `hasEvent` property defines an event which is provided within the document (to be able to get appropriate observation). Whenever the event is generated observation reasoning rules assigned to this type of event are triggered. The `represents` property references a resource, which is stored in observations about learner, after an event is generated as well.

FOHM introduces *context* and *behavior* objects. Filtering and contextual restrictions maintained by the *context* objects in FOHM is substituted by more richer reasoning language and rules in our approach. On the other hand, interactions and observations together with events substitute the notion of *behavior* objects.

### 3.2 Reasoning Rules

In this chapter we show how rules are employed to reason over distributed information sources (ontologies, user profile information, resource descriptions). The communication between reasoning

rules and the open information environment will take place by exchanging RDF annotations [25]. Rules are encoded in the TRIPLE rule language (see section 2.1).

In the following, we provide a set of rules that can be used to construct an *example*-relation between resources. Assume a user *U* is visiting some page *D*. An example, illustrating the content of this page, can be found by comparing the concepts explained on the current page with the concepts shown on an example page. Several grades of how good an example is can be derived.

The easiest way for deriving an example-relation to a page *D* is by ensuring that each concept on *D* is covered by the example *E*:

```
FORALL D, E example(D,E) <-
  studyMaterial(D) AND example(E) AND
  EXISTS C1 (D[dc:subject->C1]) AND
  FORALL C2 (D[dc:subject->C2] -> E[dc:subject->C2]).
```

The second line in the rule above ensures that *D* is **StudyMaterial** and *E* is an **Example** (according to the ontology of documents "docs"). The third rule is verifying that *D* really is about some measurable concept - thus there exists a metadata annotation like **dc:subject**. The fourth line then really expresses what our rule should check: Whether each concept on *D* will be explained in the example *E*.

Another possibility is to provide relations to examples that cover exactly the same concepts as a page *D*:

```
FORALL D, E exact_example(D,E) <-
  studyMaterial(D) AND example(E) AND
  EXISTS C1 (D[dc:subject->C1]) AND
  FORALL C1 (D[dc:subject->C1] -> E[dc:subject->C1]) AND
  FORALL C2 (E[dc:subject->C2] -> D[dc:subject->C2]).
```

The second and third line in this rule are the same as in the previous rule. The fourth and fifth line ensure that each concept on *D* is covered on *E* and vice versa.

If we want to show examples which might illustrate only some aspects of a page *D*, we can derive relations to *weaker* examples by

```
FORALL D, E weaker_example(D,E) <-
  studyMaterial(D) AND example(E) AND
  EXISTS C (D[dc:subject->C] AND E[dc:subject->C]).
```

which is be valid whenever at least on concept explained on *D* is part of the example *E*.

From the area of adaptive hypermedia, several methods and techniques have been provided to adapt the navigation and / or the content of a hyperspace to the needs, preferences, goals, etc. of each individual user. We can think of a personalized *pedagogical* recommendation of examples: The best example is an example that shows the new things to learn in context of already known / learned concepts: This would embed the concepts to learn in the previous learning experience of a user. The rule for derive this *best\_example* is as follows:

```
FORALL D, E, U best_example(D,E,U) <-
  studyMaterial(D) AND example(E) AND user(U) AND example(D,E) AND
  FORALL C ( (E[dc:subject->C] AND NOT D[dc:subject->C]) ->
    p_obs(C, U, Learned) ).
```

The rule for determining whether a user has learned some concept *C* (**p\_obs(C, U, Learned)**) is derived by checking the characteristics of the user profile. A concept is assumed to be learned if we find a **Performance** of this user via the user profile, which is related to the concept in question.

```
FORALL C, U p_obs(C, U, Learned) <- user(U) AND concept(C) AND
  EXISTS P (U[learner:hasPerformance->P] AND user_performance(P) AND
  P[learner:learningCompetency->C]).
```

The results of these rules (on the RDF-annotated and to triple translated resources provided in the Appendix) is e.g. that a page on "objects in Java (object.html)" can be related to pages which show "concepts of object orientation in Java (practical.html)" or "objects and methods in Java (objects\_methods.html)". These relations are derived by using the general "example"-rule:

```
D = sun_java:'java/concepts/object.html', E = sun_java:'java/concepts/practical.html'
D = sun_java:'java/concepts/object.html', E =
kbs_java:'java_script/examples/objects_methods.html'
```

The "exact\_example-rule" from above derives for this data set that only the "overview on object-orientation in Java (OO\_overview.html)" has an exact matching example.

```
D = kbs_java:'java_script/concepts/OO_overview.html',
E = sun_java:'java/concepts/practical.html'
```

The "weaker\_example-rule" suggest the same example page (practical.html) which exactly fits to the document OO\_overview.html also to pages about only some aspects like "methods in Java (message.html).

```
D = sun_java:'java/concepts/message.html',
E = sun_java:'java/concepts/practical.html'
```

The "best\_example" for a user who is currently visiting a page on "methods in Java (message.html)" and who has already knowledge about "objects in java" is an example illustrating these two concepts (object\_methods.html). In the data set provided in the appendix, user2 is currently in this position.

```
D = sun_java:'java/concepts/message.html',
E = kbs_java:'java_script/examples/objects_methods.html',
U = user:user2
```

Further rules for generating personalized hypertext associations can be used by more extensive use of facts from domain, user, and observation ontology. E.g. the mentioned `subConceptOf` relationship in the concept-ontology of the java application domain can be for example utilized to recommend either more general documents introducing a concept of programming strategies in general, or to recommend more specific documents (resources) about object oriented programming strategy based on requirements, level of knowledge, or interest of a user.

Sequencing relationship is another relationship which can be used to recommend documents. A document (resource) which describes a concept (the concept appears in `dc:subject` slot in meta-data about the document) from the beginning of the sequence will be recommended sooner than a document which describes a concept from the end of such a sequence.

A dependency relationship referring to whether a concept depends on another concept can be used as well. It can be used to recommend documents which describe dependent concepts together with a document describing a concept which was recommended by another rule.

## 4 Related Work

Related work to our approach can be found in the areas of open hypermedia, adaptive hypermedia, and reasoning for the semantic web. Open hypermedia is an approach to relationship management and information organization for hypertext-like structure servers. Key features are the separation of relationships and content, the integration of third party applications, and advanced hypermedia data models allowing, e.g., the modeling of complex relationships. In open hypermedia, data models like FOHM (Fundamental Open Hypertext Model) [22] and models for describing link exchange formats like OHIF (Open Hypermedia Interchange format) [14] have been developed. The use of ontologies for open hypermedia has e.g. been discussed in [19]. Here, an ontology is employed that clarifies the relations of resources. On base of this ontology, inference rules can derive new hypertext relations. In [27] the open hypermedia structures are used as an interface to ontology browsing. The links at the

user interface are transformed to queries over ontology. Thus links serves as contexts for particular user.

The question whether conceptual open hypermedia is the semantic web has been discussed in [2]. In [7], a *metadata space* is introduced, where the openness of systems and their use of metadata is compared. On the *metadata dimension* (x-axis), the units are the use of *keywords*, *thesauri*, *ontologies*, and *description logic*. The y-axis describes the *openness dimension* of systems starts from *CD ROM / file system*, *Internet*, *Web*, and ends with *Open* systems. Our approach can be seen as employing reasoning capabilities for Web-resources, or, concrete, to be on the crossings of description logic in the metadata dimension and Web in the openness dimension.

Adaptive hypermedia has been studied normally in closed worlds, i.e. the underlying document space / the hypermedia system has been known to the authors of the adaptive hypermedia system at design time of the system. As a consequence, changes to this document space can hardly be considered: A change to the document space normally requires the reorganization of the document space (or at least some of the documents in the document space). To open up this setting for dynamic document or information spaces, approaches for so called *open corpus adaptive hypermedia systems* have been discussed [5, 16]. Our approach to bring adaptive hypermedia techniques to the web therefore contribute to the open corpus problem in AH. The relation of adaptive hypermedia and open hypermedia has for example been discussed in [1].

In our approach, we used several ontologies for describing the features of *domains*, *users*, and *observations*. Compared to the components of adaptive hypermedia systems [17], an ontology for adaptive functionality is missing. However, such an ontology can be derived using the "updated taxonomy of adaptive hypermedia technologies" in [5]. Reasoning over these distributed ontologies is enabled by the RDF-querying and transformation language TRIPLE. First thoughts on using RDF in open environments have been investigated in [10, 11]. Related approaches in the area of querying languages for the semantic web can be found, e.g., in [6]. Here, a rule-based querying and transformation language for XML is proposed. A discussion of the interoperability between Logic programs and ontologies (coded in OWL or DAML+OIL) can be found in [15].

Reasoning in open worlds like the semantic web is not fully explored yet, sharing and reusing of resources with high quality is still an open problem. In this paper, we discussed first ideas on the application of rules and rule-based querying and transformation language for the domains of open hypermedia and adaptive hypermedia.

## 5 Conclusion and Further Work

In this paper, we have proposed an approach for dynamically generating personalized hypertext relations powered by reasoning mechanisms over distributed RDF annotations. We have shown an example set of reasoning rules that decide for personalized relations to example pages given some page. Several ontologies have been used which correspond to the components of an adaptive hypermedia system: a domain ontology (describing the document space, the relations of documents, and concepts covered in the domain of this document space), a user ontology (describing learner characteristics), and an observation ontology (modeling different possible interactions of a user with the hypertext). For generating hypertext structures, a presentation ontology has been introduced.

In further work, we plan to investigate to employ further ontologies like an ontology for educational models. This will enable us to add additional rules to enhance adaptive functionality based on the facts modeled in the knowledge-base by utilizing additional relationships. At the application level, we also would like to experiment with different visualization strategies for displaying results of reasoning.

## References

- [1] C. Bailey, W. Hall, D. Millard, and M. Weal. Towards open adaptive hypermedia. In *Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)*, Malaga, Spain, May 2002.
- [2] S. Bechhofer, L. Carr, C. Goble, and W. Hall. Conceptual open hypermedia = the semantic web? In *Second International Workshop on the Semantic Web*, Hong Kong, China, 2001.

- [3] T. Berners-Lee. The semantic web - mit/lcs seminar. <http://www.w3c.org/2002/Talks/09-lcs-sweb-tbl/>.
- [4] P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996.
- [5] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–100, 2001.
- [6] F. Bry and S. Schaffert. A gentle introduction into xcerpt, a rule-based query and transformation language for xml. In *International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, Sardinia, Italy, June 2002.
- [7] L. Carr, S. Bechhofer, C. Goble, and W. Hall. Conceptual linking: Ontology-based open hypermedia. In *Proceedings of the Tenth International World Wide Web Conference*, Hongkong, May 2001.
- [8] DAML+OIL, 2001. <http://www.daml.org/2001/03/daml+oil-index.html>.
- [9] P. Dolog and M. Bieliková. Towards variability modelling for reuse in hypermedia engineering. In Y. Manolopoulos and P. Návrát, editors, *Proc. of 6th East-European Conference on Advances in Databases and Information Systems — ADBIS'2002*, pages 388–400, Bratislava, Slovakia, Sept. 2002. Springer LNCS 2435.
- [10] P. Dolog, R. Gavriiloae, W. Nejdl, and J. Brase. Integrating adaptive hypermedia techniques and open rdf-based environments. In *International World Wide Web Conference*, Budapest, Hungary, May 2003.
- [11] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Towards an adaptive semantic web. Technical report, University of Hannover, July 2003.
- [12] P. Dolog and W. Nejdl. Benefits and challenges of the semantic web for user modelling. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, Budapest, Hungary, 2003.
- [13] Dublin core. <http://dublincore.org/>.
- [14] K. Gronbaek, L. Sloth, and N. O. Bouvin. Open hypermedia as user controlled meta data for the web. In *Ninth International World Wide Web Conference*, pages 554–566, Amsterdam, The Netherlands, 2000.
- [15] B. N. Grosf, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Twelfth International World Wide Web Conference*, Budapest, Hungary, May 2003.
- [16] N. Henze and W. Nejdl. Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12, 2001.
- [17] N. Henze and W. Nejdl. Logically characterizing adaptive educational hypermedia systems. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, Budapest, Hungary, 2003.
- [18] IEEE. IEEE P1484.2/D7, 2000-11-28. draft standard for learning technology. public and private information (papi) for learners (papi learner). Available at: <http://ltsc.ieee.org/wg2/>. Accessed on October 25, 2002.
- [19] S. Kampa, T. Miles-Board, L. Carr, and W. Hall. Linking with meaning: Ontological hypertext for scholars. Technical report, University of Southampton, 2001. [citeseer.nj.nec.com/kampa01linking.html](http://citeseer.nj.nec.com/kampa01linking.html).
- [20] O. Lassila and R. Swick. W3c resource description framework (rdf) model and syntax specification. Available at: <http://www.w3.org/TR/REC-rdfsyntax/>. Accessed on October 25, 2002.
- [21] D. E. Millard, L. Moreau, H. C. Davis, and S. Reich. FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains. In *Proceedings of the eleventh ACM on Hypertext and hypermedia*, pages 93–102. ACM Press, 2000.
- [22] D. E. Millard, L. Moreau, H. C. Davis, and S. Reich. FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains. In *11th ACM Conference on Hypertext and Hypermedia*, pages 93–102, San Antonio, Texas, USA, 2000.
- [23] M. Nilsson. Ims metadata rdf binding guide. <http://kmr.nada.kth.se/el/ims/metadata.html>, May 2001.
- [24] OWL, 2003. <http://www.w3.org/2001/sw/WebOnt/>.
- [25] Resource Description Framework (RDF) Schema Specification 1.0, 2002. <http://www.w3.org/TR/rdf-schema>.
- [26] M. Sintek and S. Decker. Triple - an rdf query, inference, and transformation language. In I. Horrocks and J. Hendler, editors, *International Semantic Web Conference (ISWC)*, pages 364–378, Sardinia, Italy, 2002. LNCS 2342.
- [27] M. J. Weal, G. V. Hughes, D. E. Millard, and L. Moreau. Open hypermedia as a navigational interface to ontological information spaces. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 227–236. ACM Press, 2001.
- [28] XML: extensible Markup Language, 2003. <http://www.w3.org/XML/>.

## Appendix: Set of Rules for Deriving Relations between Information Pages and Examples

```
daml := "http://www.daml.org/.../daml+oil#".
rdf := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".
doc := "http://www.example.org/doc#".
results := "http://www.results.org/results#".
sun_java := "http://java.sun.com/docs/books/tutorial/".
kbs_java := "http://www.kbs.uni-hannover.de/".
java := "http://www.kbs.uni-hannover.de/~henze/java.rdf#".

@results:data{
sun_java:'index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
sun_java:'java/index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
sun_java:'java/concepts/index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
sun_java:'java/concepts/object.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Object'].
sun_java:'java/concepts/message.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Method'].
sun_java:'java/concepts/class.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Class'].
sun_java:'java/concepts/inheritance.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Inheritance'].
sun_java:'java/concepts/interface.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Interface'].
sun_java:'java/concepts/practical.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:Example;
  dc:subject->java:'OO_Object';
  dc:subject->java:'OO_Method';
  dc:subject->java:'OO_Class';
  dc:subject->java:'OO_Inheritance';
  dc:subject->java:'OO_Interface'].

kbs_java:'java_script/examples/objects_methods.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:Example;
  dc:subject->java:'OO_Object';
  dc:subject->java:'OO_Method'].
kbs_java:'java_script/concepts/OO_overview.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Object';
  dc:subject->java:'OO_Method';
  dc:subject->java:'OO_Class';
  dc:subject->java:'OO_Inheritance';
  dc:subject->java:'OO_Interface'].

java:'OO_Object'[rdf:type->doc:Concept;
  doc:isPrerequisiteFor->java:'OO_Method'].

java:'OO_Method'[rdf:type->doc:Concept;
  doc:isPrerequisiteFor->java:'OO_Class'].

java:'OO_Class'[rdf:type->doc:Concept;
```

```

doc:isPrerequisiteFor->java:'00_Inheritance'].

java:'00_Inheritance' [rdf:type->doc:Concept;
doc:isPrerequisiteFor->java:'00_Interface'].

user:user1[
  rdf:type -> learner:Learner;
  learner:hasPerformance -> user:user1P].

user:user1P[
  rdf:type->learner:Performance].

user:user2[
  rdf:type -> learner:Learner;
  learner:hasPerformance -> user:user2P].

user:user2P[
  rdf:type->learner:Performance;
  learner:learningCompetency -> java:'00_Object'].
}

@results:simple{

  FORALL O,P,V O[P->V] <-
    O[P->V]@results:data.

  FORALL D document(D) <- D[rdf:type->doc:Document].
  FORALL C concept(C) <- C[rdf:type->doc:Concept].
  FORALL U user(U) <- U[rdf:type->learner:Learner].
  FORALL P user_performance(P) <- P[rdf:type->learner:Performance].
  FORALL E example(E) <- document(E) AND
    E[doc:hasDocumentType->doc:Example].
  FORALL E studyMaterial(E) <- document(E) AND
    E[doc:hasDocumentType->doc:StudyMaterial].

  FORALL C, U p_obs(C, U, Learned) <- user(U) AND concept(C) AND
    EXISTS P (U[learner:hasPerformance->P] AND user_performance(P) AND
    P[learner:learningCompetency->C]).

  FORALL D, E example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C1 (D[dc:subject->C1]) AND
    FORALL C2 (D[dc:subject->C2] -> E[dc:subject->C2]).

  FORALL D, E exact_example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C1 (D[dc:subject->C1]) AND
    FORALL C1 (D[dc:subject->C1] -> E[dc:subject->C1]) AND
    FORALL C2 (E[dc:subject->C2] -> D[dc:subject->C2]).

  FORALL D, E weaker_example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C (D[dc:subject->C] AND E[dc:subject->C]).

  FORALL D, E, U best_example(D,E,U) <-
    studyMaterial(D) AND example(E) AND user(U) AND example(D,E) AND
    FORALL C ( (E[dc:subject->C] AND NOT D[dc:subject->C]) ->
    p_obs(C, U, Learned) ).

```

```
}
```

```
/* Several Views */
```

```
FORALL D, E <- example(D, E)@results:simple.
```

```
FORALL D, E <- exact_example(D, E)@results:simple.
```

```
FORALL D, E <- weaker_example(D, E)@results:simple.
```

```
FORALL D, E, U <- best_example(D, E, U)@results:simple.
```