

Ontologies

Ontologies serve as a means for establishing a conceptually concise basis for communicating knowledge for many purposes. They are a popular research topic in Artificial Intelligence, e.g. knowledge engineering, natural language processing, intelligent information integration and multi-agent systems. Recently, ontologies are also applied outside the AI world, e.g. in the World Wide Web community that try to represent machine-processable data on the web using ontologies and formal metadata. The aim of this workshop is to enhance the discussion between researchers studying ontologies from different viewpoints

We wish to express our appreciation to all the authors of submitted papers, to the members of the program committee, and to the additional reviewers for making the KI-2001 “Ontologies” workshop a valuable contribution to the knowledge processing research field.

July 2001

Gerd Stumme
Alexander Mädche
Steffen Staab

Organization

The Ontologies Workshop is organized as a workshop within the 24th German/9th Austrian Conference on Artificial Intelligence. It was held on September 18, 2001, in Vienna, Austria.

Workshop Chairs

Gerd Stumme
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren (AIFB)
Universität Karlsruhe
D-76128 Karlsruhe, Germany
www.aifb.uni-karlsruhe.de/WBS/gst
stumme@aifb.uni-karlsruhe.de

Alexander Mädche
Forschungszentrum Informatik
Forschungsbereich WIM
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe, Germany
<http://www.fzi.de/wim/>
wim@fzi.de

Steffen Staab
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren (AIFB)
Universität Karlsruhe
D-76128 Karlsruhe, Germany
www.aifb.uni-karlsruhe.de/WBS/sst
staab@aifb.uni-karlsruhe.de

Program Committee

Andreas Abecker
(*DFKI, Kaiserslautern*)

Jürgen Angele
(*Ontoprise GmbH, Karlsruhe*)

Dieter Fensel
(*Free University of Amsterdam*)

Christopher Habel
(*Universität Hamburg*)

Udo Hahn
(*Universität Freiburg*)

Heinrich Herre
(*Universität Leipzig*)

Ulrich Reimer
(*Swiss Life, Zürich*)

Heiner Stuckenschmidt
(*TZI Bremen*)

Rudi Studer
(*Universität Karlsruhe*)

Table of Contents

Full Papers

Ontologie Structures in CoMet	1
<i>J. Yli-Koivisto, J. Puustjarvi</i>	
Ontological Engineering for Conceptual Modeling	16
<i>R. Raban and B. Garner</i>	
Contributions to the Axiomatic Foundations of Upper-Level Ontologies ...	30
<i>W. Degen and H. Herre</i>	
From Domain Ontologies to Object-Oriented Frameworks	41
<i>G. Guizzardi and R. de Almeida Falbo and J. Gomcalves Pereira Filho</i>	

Position Papers

Ontological Aspects in Representing Mathematical Knowledge for Reasoning and Presentation Purposes	55
<i>H. Horacek</i>	
Ontologies, Multi-Perspective Modelling and Knowledge Auditing	60
<i>J. Kingston</i>	
The Ontology of Commonsense Knowledge	66
<i>W.S. Saba</i>	
EOS: Making the Epistemic Impact of Ontologies in Knowledge Processing Explicit	81
<i>W. Wohner</i>	

Application Papers

An Ontology for WWW Summarization in Bone Marrow Transplantation (BMT): An Interim Report	94
<i>B. Endres-Niggemeyer and B. Hertenstein and C. Villiger and C. Ziegert</i>	
Building Shared Ontologies for Terminology Integration	105
<i>G. Schuster and H. Stuckenschmidt</i>	

Ontological Engineering for Conceptual Modeling

Ryszard (Richard) Raban¹, Brian Garner²

¹Faculty of Information Technology
University of Technology, Sydney
richard@soecs.uts.edu.au

²School of Computing and Mathematics
Deakin University
brian@deakin.edu.au

Abstract. In acknowledging the importance of ontologies in conceptual modeling, database integration and business process modeling, this paper introduces a set of principles for building ontologies. Starting from Guarino's meta-properties of ontological terms, the paper describes the denotational semantics of the meta-properties and derives from them some engineering rules and checks for constructing domain specific conceptual models, based on the overarching requirement to assign meanings to concepts using tags and labels. Parallel research by the authors into the use of contextual references and roles to restrict such meanings will be published elsewhere.

1 Introduction

Ontologies grew out of being a philosophical endeavor of finding the top level categories of existence (those appear in the works of Aristotle through Kant, Heidegger to the ontological categories introduced recently by Sowa [1]) into an important tool in domain knowledge representation, and recently, in information systems modeling. Whenever a model is built, a certain ontological commitment is assumed. In most cases, however, this commitment is not explicitly stated. It makes information exchange, interoperability and integration very difficult. XML [2], XMI [3] or UML [4], [5], [6], to mention just some of the prominent modeling and integration tools, will not be able to realize their full potential in the area of knowledge sharing and integration unless the tags and labels used have ontology-based semantics.

As ontologies become more and more part of knowledge engineering and modeling [7], [8], [9], it is imperative to have precise rules for engineering ontologies themselves.

A similar problem also arises in managing changing contexts within reasoning processes which, as argued in [10], often requires dynamic ontology modifications. In order to be able to manipulate ontologies without losing their internal consistency, strict ontology construction rules are required.

This paper aims to advance the basis of ontological engineering by specifying rules for creating a hierarchy of properties that individuals might have and by exploring some implications of the rules for conceptual modeling. We start by defining the

denotational semantics of ontological meta-properties, also referred to in this paper as criteria, and then derive from them rules for properties subtyping. The meta-properties, which were partially introduced in [11], [12], [13], [14], [15] and have been recently refined and formalized by Guarino and Welty in [16], were adopted as the starting point for this research.

2 Extensional semantics of meta-properties.

Let's assume that in a particular Universe of Discourse there are individuals x that can be dynamically created and destroyed.

Let ω_i be the existence predicate defined on all x such that $\omega_i(x)$ means that an individual x exists at a point in time t . We assume that each individual might have exactly one creation time t_{cx} such that

$$\forall_{t < t_{cx}} \neg \omega_i(x) \wedge \exists t_{\Delta} \forall_{t_{cx} \leq t < t_{cx} + t_{\Delta}} \omega_i(x)$$

and exactly one destruction time t_{dx} such that

$$\forall_{t > t_{dx}} \neg \omega_i(x) \wedge \exists t_{\Delta} \forall_{t_{dx} - t_{\Delta} < t \leq t_{dx}} \omega_i(x).$$

Let $\tau_x = [t_{cx}, t_{dx}]$ be the period in which an individual x exists or the individual x 's *lifetime*.

Let $\Omega_t = \{x \mid \omega_i(x)\}$ be a set of all individuals existing at a point in time t .

First-order properties correspond to monadic predicates over Ω_t . If P is a property, $P(x)$ means that an individual x has property P at a point in time t . This, of course, implies that the individual x must exist at the point in time t to have the property. For example, individuals of a Universe of Discourse might have properties like *PERSON*, *EMPLOYEE*, *RED*, *SHORT*, etc. Note that properties are not attributes like *LENGTH*, or *COLOR* which are functions defined on a domain of individuals and return attribute values. For example, $LENGTH(x) = 175\text{cm}$ or $COLOR(y) = \text{'red'}$.

An individual x acquires a property P at a point in time $t_{cP(x)}$ such that

$$\forall_{t < t_{cP(x)}} \neg P_i(x) \wedge \exists t_{\Delta} \forall_{t_{cP(x)} \leq t < t_{cP(x)} + t_{\Delta}} P_i(x)$$

and discards it at a point in time $t_{dP(x)}$ such that

$$\forall_{t > t_{dP(x)}} \neg P_i(x) \wedge \exists t_{\Delta} \forall_{t_{dP(x)} - t_{\Delta} < t \leq t_{dP(x)}} P_i(x).$$

A period of time during which an individual x holds a property P is called an *attribution period* of P to x and is defined as $\tau_{P(x)} = [t_{cP(x)}, t_{dP(x)}]$ such that

$$\forall_{t \in \tau_{P(x)}} P_i(x).$$

In this paper, the discussion is limited to such properties that apply for some period of time to at least one individual, that is $\forall_P \exists_x t_{cP(x)} < t_{dP(x)}$.

A property P can have many attribution periods for an instance x . For example, a person can hold property *STUDENT* many times in different periods of time.

We call a property P *static* if;

$$\forall_x P_i(x) \rightarrow (\tau_{P(x)} = \tau_x)$$

Otherwise, a property is called *dynamic*.

A static property is inherent to an individual; it is acquired at its creation time and holds for its entire lifetime. Properties like *PERSON*, *LIVING-BEING* are static. A dynamic property is acquired by an individual temporarily, and it can be acquired and discarded many times in an individual's lifetime. The previously mentioned property *STUDENT* is an example of a dynamic property.

The denotation of a property P is defined as $\delta P_t = \{x \mid P_t(x)\}$ and represents a set of all individuals that have property P at a point in time t .

It is also valid to use the denotation for the existence predicate - $\delta\omega_t$ is a set of all individuals existing at a point in time t .

2.1 Subtyping

If a property Q is a subtype of a property, denoted by $Q \leq P$, then

$$\forall_t (\delta Q_t \subseteq \delta P_t).$$

If additionally

$$\exists_t (\delta Q_t \subset \delta P_t),$$

then Q is a *proper subtype* of P , which we denote by $Q < P$. If a property Q is a (proper) subtype of a property P , it can be said that a property P is a (*proper*) *supertype* of a property Q .

There could be two different types of subtyping. Firstly, we can have time independent or *static subtyping*. In this case, a static property Q is a subtype of a static property P . For example, property *PERSON* is a static subtype of property *LIVING-BEING* and it means that a *living-being* can also be a *person*, and if it is so, it remains a *person* for its lifetime.

Secondly, we can have time dependent subtyping or *dynamic subtyping*. In this case, a dynamic property Q is a subtype of a property P , which could be either static or dynamic. It means that an individual x with the property P might have the property Q at some periods of time and might not have this property on some other occasions. For example, property *STUDENT* is a dynamic subtype of property *PERSON* as a *person* might be a *student* only for some periods of time. Similarly, properties *PART-TIME-STUDENT* and *FULL-TIME-STUDENT* are dynamic subtypes of property *STUDENT* and here too a *student* can change its status from being a *part-time-student* to being a *full-time-student*, and vice-versa, many times during the period of enrolment.

Using the above terminology let's define the semantics of the three meta-properties introduced by Guarino [16]: identity, rigidity and dependence.

2.2 Identity

Following the original definition, we say that a property P has *identity*₁ and denote it by +I, if there is a relation R such that

$$\forall_t \forall_{xy} (P_t(x) \wedge P_t(y)) \rightarrow (R(x,y) \leftrightarrow (x = y)).$$

The relation R is called an identity condition of P . For example, property *PERSON* has identity as its instances can be distinguished from each other by their DNA structure (relation *HAS-SAME-DNA*) or the makeup of their brains (relation *HAS-SAME-BRAIN*). On the other hand, individuals of property *THING* cannot be distinguished by any identity condition inherent to all things.

Further we say that a property P has its own identity, and denote it by +O, if there is a relation R such that

$$\forall_i \forall_{xy} (P_i(x) \wedge P_i(y)) \rightarrow (R(x,y) \leftrightarrow (x = y))$$

and

$$\forall_Q (\neg(Q < P)) \rightarrow (\neg(\forall_i \forall_{xy} (Q_i(x) \wedge Q_i(y)) \rightarrow (R(x,y) \leftrightarrow (x = y)))).$$

In other words, a property P has its own identity, if it does not share its identity condition defined by the relation R with any property Q which is not a subsumption of the property P . For example, relation *HAS-SAME-DNA* allows us to distinguish between any two individuals having property *LIVING-BEING*, but it cannot be used to distinguish between individuals having a property being a supertype of *LIVING-BEING*. Thus, property *LIVING-BEING* has its own identity.

However, property *PERSON* shares its identity condition *HAS-SAME-DNA* with all individuals with property *LIVING-BEING*. And unless property *PERSON* introduces its own identity condition, we say that the property does not have its own identity, and denote it by -O.

Obviously, if a property has its own identity (+O) it also has identity (+I). Conversely, not having identity (-I) rules out its own identity, and therefore, implies lack of own identity (-O). This leaves only three possible identity criteria: -I-O, +I-O, and +I+O.

Let's explore under what conditions a subtype Q of a property P with given identity criteria can inherit or change the identity criteria of its supertype.

If a property P has identity, it means that there is an identity condition defined for its individuals. For any subtype property Q of P , individuals with property Q have also property P and, therefore, can be distinguished using the property P 's identity condition. Thus, having identity is always inherited.

If a subtype property Q of P has its own identity, it becomes +I+O irrespective of what kind of identity criteria the supertype has. If, however, a subtype property Q of P does not have its own identity, it always becomes +I-O, unless its supertype does not have identity at all.

These subtyping rules for identity are summarized in TABLE I. The boxes with the phrase 'not allowed' signify the fact that identity cannot ever be lost in subtypes or acquired from a subtype, which does not have identity. The symbol $\mathbf{I}(\top)$ means that subtyping is allowed under the identity criteria, and the symbol $\mathbf{I}(\perp)$ means that subtyping is not allowed under the identity criteria.

TABLE I. Subtyping Rules for Identity.

$Q \leq P$	-I-O		+I-O		+I+O	
-I-O	allowed	$\mathbf{I}(\top)$	not allowed	$\mathbf{I}(\perp)$	not allowed	$\mathbf{I}(\perp)$
+I-O	not allowed	$\mathbf{I}(\perp)$	allowed	$\mathbf{I}(\top)$	allowed	$\mathbf{I}(\top)$
+I+O	allowed	$\mathbf{I}(\top)$	allowed	$\mathbf{I}(\top)$	allowed	$\mathbf{I}(\top)$

2.3 Rigidity

A property P is *rigid*, which is denoted by +R, if it is a static property. It means that all individuals with property P meet the rigidity condition

$$P_i(x) \rightarrow (\tau_{P(x)} = \tau_x).$$

A property is *non-rigid*, which is denoted by -R, if

$$\exists_{i'} \exists_{i''} \exists_x (P_{i'}(x) \wedge \omega_{i''}(x) \wedge \neg P_{i''}(x)).$$

It means that there is at least one individual that violates the rigidity condition at some point in time.

A property is *anti-rigid*, which is denoted by ~R, if

$$\exists_{i'} \exists_{i''} \forall_x (P_{i'}(x) \rightarrow (\omega_{i''}(x) \wedge \neg P_{i''}(x))).$$

Anti-rigidity is a special case of non-rigidity, in which all individuals holding a property P violate the rigidity condition, and as such is subsumed by non-rigidity.

A rigid property P can have a rigid subtype property Q , if Q holds for its individuals for their entire lifetime. For example, property *LIVING-BEING* and its subtype *PERSON*. This amounts to static subtyping. But also, a subtype property Q of a rigid property P can be a result of dynamic subtyping, and then it can be either non-rigid or anti-rigid, depending whether some or all of its individuals violate the rigidity condition. Therefore, rigidity is not inherited.

It is possible to have a rigid proper subtype Q of a non-rigid property P , since the denotation $\delta Q_i \subset \delta P_i$ could contain only those individuals out of δP_i , which fulfill the rigidity condition. Equally, it is possible that a proper subtype Q of a non-rigid property P has the denotation $\delta Q_i \subset \delta P_i$ that contains only those individuals out of δP_i , which violate the rigidity condition, or contains some that do and some that do not. Hence, non-rigidity is not inherited.

Since all individuals that have an anti-rigid property P hold it for periods of time always shorter than their lifetimes, any subtype property Q of the property P cannot hold it longer than P 's lifetime. Therefore, all individuals of any subtype property Q must violate the rigidity condition, which means that anti-rigidity is inherited.

The summary of subtyping rules of the rigidity conditions is presented in TABLE II. The symbol $\mathbf{R}(\mathbf{T})$ means that subtyping is allowed under the rigidity criterion, and the symbol $\mathbf{R}(\perp)$ means that subtyping is not allowed under the rigidity criterion.

TABLE II. Subtyping Rules for Rigidity

$Q \leq P$	+R	-R	~R
+R	allowed $\mathbf{R}(\mathbf{T})$	allowed $\mathbf{R}(\mathbf{T})$	not allowed $\mathbf{R}(\perp)$
-R	allowed $\mathbf{R}(\mathbf{T})$	allowed $\mathbf{R}(\mathbf{T})$	not allowed $\mathbf{R}(\perp)$
~R	allowed $\mathbf{R}(\mathbf{T})$	allowed $\mathbf{R}(\mathbf{T})$	allowed $\mathbf{R}(\mathbf{T})$

2.4 Dependence

Following Guarino, we confine this discussion to one type of dependence that is “*notional dependence*, which holds for a property if its instances require instances of another property to exist” [16]. We say that a property P is *dependent*, which is denoted by +D, if

$$\forall_i \forall_x (P_i(x) \rightarrow \exists_{Q \neq P} \exists_{y \neq x} Q_i(y)).$$

In other words, for an individual x to have a property P , it is required that there exists an individual having a property Q . In this definition, it is also assumed that Q is not a part of P . For example, individuals with property *PARENT* require individuals with property *CHILD* to exist.

A property P is *independent*, which is denoted by -D if it not dependent.

If a property Q is a subtype of a dependent property P , all individuals with the property Q also have the property P , and are therefore subject to the same dependency condition, and as such, are dependent. Thus, dependence is inherited by subtypes. On the other hand, an independent property P can have dependent or independent subtypes.

The summary of subtyping rules of the dependence criterion is presented in TABLE III. The symbol $\mathbf{D}(\top)$ means that subtyping is allowed under the dependency criterion, and the symbol $\mathbf{D}(\perp)$ means that subtyping is not allowed under the dependency criterion.

TABLE III. Subtyping Rules for Dependence

$Q \leq P$	+D	-D
+D	Allowed $\mathbf{D}(\top)$	allowed $\mathbf{D}(\top)$
-D	not allowed $\mathbf{D}(\perp)$	allowed $\mathbf{D}(\top)$

3 Ontological Engineering Rules

All desirable combinations of the three property criteria give rise to eight property types (a metalevel classification of properties), which were described in [16]. Three of them are called Formal as they do not carry identity, and the remaining five are called sortals as they have either their own or an inherited identity. The property types are shown in TABLE IV. Note that we have split Category, Type and Merely Rigid Sortals according to their dependency criterion and Material Role according to its own identity criterion. This will enable us to have a closer look at subtyping allowed for those types of properties.

TABLE IV. Property Types Resulting from Property Criteria.

	Property Type	Property Criteria	Examples
F O R M A L	Category+ (Cat+)	-O-I+D+R	<i>SOCIAL-ENTITY</i>
	Category- (Cat-)	-O-I-D+R	<i>THING, LOCATION, ENTITY</i>
	Formal Role (FRole)	-O-I+D~R	<i>PART, PATIENT, ACTOR</i>
	Atribution (Attr)	-O-I-D-R	<i>MALE, RED</i>
S	Type-Atribution Mixing (ATMix)	-O+I-D-R	<i>MALE-PERSON, RED-FLOWER</i>
O R T	Type+ (Type+)	+O+I+D+R	<i>ORGANIZATION</i>
	Type- (Type-)	+O+I-D+R	<i>LIVING-BEING</i>
	Phasal Sortal (PSort)	+O+I-D~R	<i>CATERPILAR</i>
	Material Role+ (MRole+)	+O+I+D~R	<i>STUDENT</i>
A	Material Role- (MRole-)	-O+I+D~R	<i>FOOD</i>
L S	Merely Rigid Sortal+ (MRSort+)	-O+I+D+R	<i>LORD</i>
	Merely Rigid Sortal- (MRSort-)	-O+I-D+R	<i>INVERTEBRATE-ANIMAL</i>

TABLE V summarizes the allowed subsumptions between property types shown in TABLE IV. Cells contain three subsumption conditions originating from identity, dependence and rigidity criteria discussed earlier. These are shown in the table, as appropriate, for the respective property types. For example, at the intersection of Formal Attribution (**Attr** row) and Formal Role (**FRole** column) there are:

- $\mathbf{\tau}(T)$ indicating that the identity criterion allows the subsumption, as per TABLE I.
- $\mathbf{D}(\perp)$ which indicates that the dependency criterion does not allow the subsumption, as per TABLE III.
- $\mathbf{R}(\perp)$ which indicates that the rigidity criterion does not allow the subsumption, as per TABLE II; and since it is a conjunction of the three conditions, it follows that the subsumption is not allowed. All allowed subsumptions are highlighted by shading of the cells.

3.1 Guarino's ontology engineering rules.

Let us examine subtyping rules given in [16] and assess their coverage in the discussion so far:

- G1. “*Antirigid class cannot subsume a rigid class.*” This rule has already been accounted for in the discussion of rigidity.
- G2. “*ICs cannot be ‘overridden’ by a subclass, merely augmented*” (in other words, sortals cannot subsume formal concepts). This has already been accounted for in the discussion of identity.
- G3. “*A dependent class cannot subsume an independent one.*” This has already been accounted for in the discussion of dependence.
- G4. “*A material role can be subsumed by rigid sortals, since they carry identity ... They are the roles that normally specialise formal roles...*” In the discussion so far, a material role is the most flexible of all the property types in terms what it can subsume (see TABLE V). Except for material roles without own identity, which cannot be subsumed by formal properties, all the other subsumptions are allowed. However, the cited rule makes good sense, since it would be desirable to have in the ontology some indication of what property the role applies to. Otherwise, the role specification would be somehow incomplete, like saying that there is a material role STUDENT without any indication that it applies to PERSON. But there is no reason to make the assumption that a rigid property must be subsumed by a rigid one; there could be some other material roles or even phased sortals between the material role and the subsuming rigid property in the subsumption hierarchy.
- G5. “*Phased sortals must be subsumed by a type.*” This is required to be able to establish the identity of the changing entity. But since a merely rigid, independent sortal, type-attribution mixing, material role and phased sortal should all be subsumed by a type, all of them can directly subsume a phased sortal.
- G6. “*Merely rigid sortals ... are always subsumed by at least one type.*” This is to provide an identity condition, as merely rigid sortals do not carry their own identity condition.
- G7. “*Types can only be subsumed by categories and strictly non-rigid formal attributions.*” In other words, types sit between formal properties and the rest of the sortals in the subsumption hierarchy. In fact, it has already been said that the other sortals should be eventually subsumed by a type. Of course, types can form an hierarchy and subsume each other before being subsumed by a category or formal attribution.

TABLE VI summarizes all the conditions discussed so far, and shows when subtyping of properties of different types is allowed, as indicated by the content of respective cells. If “not allowed” is written in a cell, it means that there is a restriction on the subsumption originating from TABLE V. In cases where “G(*n*) not allowed” is written, it indicates that one of the Guarino’s rule *n* has been invoked to prohibit this subsumption. For example, a formal attribution (**Attr** row) cannot be subsumed by a formal role (**FRole** column) since the cell on the intersection of the two contains phrase “not allowed” derived from TABLE V.

		F O R M A L				S O R T A L S								
\subseteq		Cat+ O-I+D+R	Cat- O-I-D+R	FRole O-I+D~R	Attr O-I-D-R	ATMix O+I-D-R	Type+ O+I+D+R	Type- O+I-D+R	PSort O+I-D~R	MRole+ O+I+D-R	MRole- O+I+D~R	RSort+ O+I+D+R	RSort- O+I-D+R	
F O R	Cat+ -O-I+D+R	allowed	allowed	not allowed	allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	
	Cat- -O-I-D+R	not allowed	allowed	not allowed	allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	
M A L	FRole -O-I+D~R	allowed	allowed	allowed	allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	
	Attr -O-I-D-R	not allowed	allowed	not allowed	allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	not allowed	
S	ATMix -O+I-D-R	not allowed	not allowed	not allowed	not allowed	allowed	not allowed	allowed	not allowed	not allowed	not allowed	not allowed	allowed	
O	Type+ +O+I+D+R	allowed	allowed	not allowed	allowed	(G7) not allowed	allowed	allowed	not allowed	not allowed	not allowed	(G7) not allowed	(G7) not allowed	
R	Type- +O+I-D+R	not allowed	allowed	not allowed	allowed	(G7) not allowed	not allowed	allowed	not allowed	not allowed	not allowed	not allowed	(G7) not allowed	
T	PSort +O+I-D~R	not allowed	(G5) not allowed	not allowed	(G4) not allowed	allowed	not allowed	allowed	allowed	not allowed	not allowed	not allowed	allowed	
A	MRole+ +O+I+D~R	(G4) not allowed	(G4) not allowed	allowed	(G4) not allowed	(G4) not allowed	allowed	allowed	allowed	allowed	allowed	allowed	allowed	
L	MRole- -O+I+D~R	not allowed	not allowed	not allowed	not allowed	(G4) not allowed	allowed	allowed	allowed	allowed	allowed	allowed	allowed	
	MRSort+ -O+I+D+R	not allowed	not allowed	not allowed	not allowed	allowed	allowed	allowed	not allowed	not allowed	not allowed	allowed	allowed	
S	MRSort- -O+I-D+R	not allowed	not allowed	not allowed	not allowed	allowed	not allowed	allowed	not allowed	not allowed	not allowed	not allowed	allowed	

4 Using Property Types in Conceptual Modeling

In knowledge bases [17], [18] and databases [19], [20], there is a commonly accepted distinction between definitions of terms, their characteristics and the relations between them (often referred to as a schema or the TBox), and assertions about the world stated in instances of those terms (often referred to as a fact base or the ABox). This section will discuss implications of the taxonomy of properties for defining domain models, i.e. classes, their characteristics and the relations between them (the TBox).

In conceptual modeling of an application domain, not only properties are given, but also all the sufficient and necessary conditions for an individual to carry the specific property are defined. These property definitions are typically called classes. Individuals that hold the property defined by a class are called instances of the class. In order to maintain the distinction between properties and classes, the former will be denoted by all capital (hyphenated) names (e.g. *LIVING-BEING*), and the latter by non-hyphenated names with the first letter of each word capitalized (e.g. *LivingBeing*). Classes based on properties will have corresponding names consisting of the same wording.

A class defines domain specific conditions for the class membership. The conditions give characteristics that are pertinent to all the class members in the context of the domain and the purpose for which the modeling is performed. Properties of type Attribution seems to provide appropriate terminology for specifying class characteristics. However, property type Attribution, as explained in [16], is

“referring ... to an attribution as the property of having an attribute with certain value, i.e. having gender MALE or color RED.”

And therefore, while useful to describe characteristic of individuals, attributions are not suitable for declaring that class Person should have gender stated for its instances. This requires a relation, which maps instances of the class to a set of attributions relevant to the characteristic being defined. The sets of attributions will be called attribution types. This brings us to the first relation type of conceptual modeling, which has the following format:

Characteristic(Class, AttributionType).

For class Person, this relation type can be instantiated as

Gender(Person, {“Male”, “Female”}) – enumerated attribution type

Height(Person, {x: x is distance measurement in meters}) – measured attribution type

Address(Person, {x: x is a character string}) – description attribution type

Dependent properties come into existence in relation to some other properties holding over different individuals. For example, *STUDENT* is a property of a person being enrolled for a university course, or *BANK-CUSTOMER* is a property of a person having a bank account. The statements have the following elements in them:

- they tell us that *STUDENT* and *BANK-CUSTOMER* are subtypes of property *PERSON*, and

- they also point out what properties *STUDENT* and *BANK-CUSTOMER* are dependent on; in this case the properties depend on the existence of respectively, a *UNIVERSITY-COURSE* and of a *BANK-ACCOUNT*.

It is easy to show that there are relationships between classes based on these properties by creating pairs (Student, UniversityCourse) and (BankCustomer, BankAccount). What remains to be answered is what type of relationships do they represent.

In order to answer this question, it is useful to call upon Charles Peirce's three basic categories Firstness, Secondness and Thirdness which were described in the quotation from the original provided in [1]:

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

Obviously, independent properties are related to the Firstness. Dependent properties relate to the Secondness. The Thirdness, which mediates the relationship, provides an answer to the original question. The mediating element that brings *STUDENT* and *UNIVERSITY-COURSE* together is *ENROLMENT*, and the one that brings *BANK-CUSTOMER* and *BANK-ACCOUNT* together is *BANK-ACCOUNT-CONTRACT*. In general, dependent properties will result in a conceptual relation type defined as follows:

DependencyMediator(DependentClass, Class)

which for the two examples is instantiated as:

Enrolment(Student, UniversityCourse)

BankAccountContract(BankCustomer, BankAccount).

The only other issue here is what property type DependencyMediator is based on. Mediating properties seems to be rigid, dependent and carry their own identity, and therefore they are of type **Type+**.

Another type of relationships that can exist between properties is the structural relationship. In this case, the relations bring about certain arrangement of individuals. For example, PartOf(Engine, Car), Above(Roof, Basement). This type of relationship can be defined as

StructuralDependency(Class, Class).

Structural dependencies are properties that seem to be both dependent and non-rigid. It is the type of properties, which are not included in the taxonomy as they are *"too weak ... to capture the rigor ... intended"* [16]. However, if a structural dependency is combined with one of the related properties, for example, *PART-OF-CAR* or *ABOVE-BASEMENT*, it becomes independent and non-rigid and therefore is of type Attribution.

5 Conclusion

This paper critically reviews the application of ontological engineering to conceptual modeling. The two activities, while related, differ in purpose. Ontological engineering deals with rules and principles of conceptualization of a problem domain. Conceptual modeling, on the other hand, is concerned with the problem domain structuring. While one may accept that no structure exists without underlying

conceptualization, the requisite conceptualization is usually implicit and often derived on an intuitive, craft basis. Manifestation of an explicit basis for conceptualization will, in our opinion, not only result in IT solutions of greater integrity, but will also facilitate system and data integration vital to enterprise application integration, including semantic web applications.

In the first part, the paper presented denotational semantics of identity, rigidity and dependence used to define types of properties, and then introduced a set of property subtyping rules and checks. It established the foundations for creating clean, well-structured taxonomies. In the second part, the links between ontology engineering principles and conceptual modeling were explored. In particular, three types of emergent relationships were discussed.

Further work will concentrate on the development of domain ontologies as the basis for data and system integration for oceanographic data interchange and electronic business processes. Parallel work on the rules for context management and role constraints will also benefit from the explication of property subtyping rules and constraints.

References

1. Sowa, J.F., *Knowledge Representation*. 2000, Pacific Grove, CA USA: Brooks/Cole. 593.
2. Connolly, D., *Extensible Markup Language (XML)*. 1997, W3C.
3. Object Management Group, *XMI Metadata Interchange (XMI)*. 1999, OMG.
4. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Object Technology Series. 1999, Reading, MA: Addison-Wesley.
5. Rumbaugh, J., I. Jacobson, and G. Booch, *The Unified Modelling Language Reference Manual*. Object Technology Series. 1999, Reading, MA: Addison-Wesley.
6. Booch, G., J. Rumbaugh, and I. Jacobson, *The Unified Modelling Language User Manual*. Object Technology Series. 1999, Reading, MA: Addison-Wesley.
7. Zhu, X., et al., *Ontology-Based Web Site Mapping for Information Exploration*, in *Proceedings of the Eighth International Conference on Information Knowledge Management*. 1999: Kansas City, MO USA. p. 188-194.
8. Embley, D.W., et al., *Ontology-based Extraction and Structuring of information from Data-Rich Unstructured Documents*, in *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management*. 1998: Washington, USA. p. 52-58.
9. Valente, A., et al., *Building and (Re)Using an Ontology of Air Campaign Planning*. IEEE Intelligent Systems, 1999. **14**(1 (Jan/Feb 1999)): p. 27-36.
10. Keyser, D., *Ontologically Yours*, in *Lectures in Artificial Intelligence*, M.-L. Mugnier and M. Chein, Editors. 1998, Springer-Verlag. p. 35-47.
11. Sowa, J.F., *Using a Lexicon of Canonical Graphs in a Semantic Interpreter*, in *Relational Models of the Lexicon*, M.W. Evens, Editor. 1988, Cambridge University Press: Cambridge.
12. Strawson, P.F., *Individuals. An Essay on Descriptive Metaphysics*. 1959, London: Routledge.
13. Griffin, N., *Relative Identity*. 1977, Oxford: Oxford University Press.
14. Guarino, N., M. Carrara, and P. Giaretta, *An Ontology of Meta-Level Categories*, in *Principles of Knowledge Representation and reasoning: Proceedings of the Fourth International Conference (KR'94)*, J. Doyle, E. Sandewall, and P. Torasso, Editors. 1994, Morgan Kaufman Publishers: San Francisco, CA.
15. Guarino, N., *Formal Ontology, Conceptual Analysis and Knowledge Representation*. International Journal of Human-Computer Studies, 1995. **43**(5/6): p. 625-640.

16. Guarino, N. and C. Welty, *A Formal Ontology of Properties*, in *Proceedings of EKAW-2000: The 12th International Conference on Knowledge Engineering and Knowledge Management*, R. Dieng and O. Corby, Editors. 2000, Springer-Verlag. p. 97-112.
17. Brachman, R.J., *An Overview of the KL-ONE Knowledge Representation System*. Cognitive Science, 1985. **9**(2): p. 171-216.
18. Levesque, H.J. and R.J. Brachman, *Knowledge Level Interfaces to Information Systems*, in *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, M.L. Brodie and J. Mylopoulos, Editors. 1986, Springer-Verlag: New York.
19. Abrial, J.R., *Data Semantics*, in *Data Base Management*, J.W. Klimbie and K.L. Koffeman, Editors. 1974, Noth-Holland Publishing: Amsterdam. p. 1-59.
20. Mylopoulos, H.J. and P.A. Bernstein, *A Language Facility for Designing Database-Intensive Applications*. ACM Journal on Database Systems, 1980. **5**(2): p. 185-207.

Contributions to the Axiomatic Foundation of Upper-Level Ontologies

Wolfgang Degen¹, Heinrich Herre²

IMMD, Universität Erlangen, degen@informatik.uni-erlangen.de¹
Institut für Informatik, Universität Leipzig, herre@informatik.uni-leipzig.de²

Abstract. In the present paper we make some contributions to the theory of upper level ontologies. Every domain-specific ontology must use as a framework some upper-level ontology which describes the most general, domain-independent categories of reality. It turns out that the top-level ontology of the well-known standard modelling languages KIF (Knowledge Interchange Format), F-logic (Frame-Logic), and CyCL, is based on set-theoretical construction principles, so that the latter are essentially limited by the extensionalism of set theory. In the current paper we outline a new approach to upper-level ontologies which is based on recent results in formal ontology. We formulate some criteria which any upper-level ontology should satisfy although we shall not give a definite definition. Every such ontology should include the three ontological categories of *sets*, *universals*, *individuals* and a system \mathcal{R} of *formal relations*. \mathcal{R} contains, among others, membership, part-whole and inheritance. The individuals are divided in substantoids, moments and situoids. We review a modelling language GOL (General Ontological Language) whose top-level ontology satisfies these criteria. Finally, we compare our top-level ontology to others, in particular to KIF, and to the ontologies of Russell-Norvig, J.Sowa, and to the standardization project SUO.

1 Introduction

The current paper is devoted to an analysis of upper level categories. Upper level ontologies are concerned with theories of such highly general (domain-independent) categories as: time, space, inheritance, instantiation, identity, processes, events, attributes, etc. The development of a widely accepted and well-established upper level ontology is an important and ambitious task, important because it determines a framework for the construction of domain-specific and generic ontologies, ambitious because it is concerned with highly debated philosophical problems. An upper level ontology could be considered as an integral part of a knowledge modelling and representation language. In our approach we assume the point of view of realism, i.e. the position that the kind of things we are speaking about have objective existence; hence, ontological research should address reality itself. Thus, ontological research is not limited to building up conceptual schemes. We believe that the real world itself plays an important role to achieve a unified and consistent top-level ontology. We use the term *category* to denote collections of entities. We classify the collection of all entities into

the categories *sets*, *individuals*, *universals*, and *formal relations*. In section 2 we study the categories of sets, individuals, and universals. Section 3 is devoted to formal relations. The categories of section 2 and section 3 present the highest level of generality. In section 4 the category of individuals is further classified and some formal relations are considered in more detail. In section 4.2 conditions are formulated that - in our opinion- should be satisfied by every upper level ontology, and in section 4.3 a formal modelling language GOL is reviewed whose ontological basis satisfies the criteria of an upper level ontology. GOL is an ongoing research project at the University of Leipzig, [10]. In section 5 we compare our ontology to other upper-level ontologies.

2 Individuals, Universals, and Sets

In our approach the entities of the real world can be partitioned in sets and urelements. Urelements are entities which are not sets, and urelements are classified into *individuals* and *universals*. Individuals are thought of as a realm of concretely existing things in the world (not sets), within the confines of space and time. An individual is a single thing located in some single region of space-time. A universal is an entity that can be instantiated by a number of different individuals which are similar in some respect. Universals are patterns of features which are not related to time and space. We assume that universals exist *in* the individuals which instantiate them (thus they exist *in re*); thus, our attitude is broadly Aristotelian in spirit. Individuals are correlated with universals via the formal relation of *instantiation*. We write $a :: u$ to denote that the individual a is an instance of the universal u . In the philosophical tradition there are several conceptions of universals and individuals, respectively, [13]. For example, universals have been conceived as *sets of individuals*; and individuals have been conceived as *sets of universals*. If one adopts both conceptions, then individuals are sets of sets of individuals; and universals are sets of sets of universals. And so on. We conclude that not both conceptions are admissible. Which of the two conceptions has to be given up? We think that both have to be given up. A detailed justification of this view is long and will be expounded in the full version of the paper.

We assume that sets are neither in Space nor in Time. Hence, sets are not individuals. A universal is determined by its intention which captures among others a certain granularity and a certain view of its instances. We assume the basic axiom that sets, individuals and universals are pairwise disjoint categories.

3 Formal Relations

According to the present standard doctrine, every n -ary relation is a set (or class) of ordered n -tuples. This conception is applied indiscriminately to both so-called empirical and mathematical relations. Thus, when John is kissing Mary, the ordered pair $\langle John, Mary \rangle$ is an element of a certain set A of ordered pairs that is denoted by "kissing". Also, that $2 < 5$ is analysed by $\langle 2, 5 \rangle \in B$ where B

is the set $\{\langle x, y \rangle : x < y\}$ of ordered pairs. This uniform analysis harmonizes with conceiving of both sentences as having the same syntactic form, namely $R(a, b)$. The usual semantics of first-order logic, i. e. the semantics with respect to which Gödel's completeness theorem is proved, interprets any sentence $R(a_1, \dots, a_n)$ as stating that $\langle a_1, \dots, a_n \rangle \in R$.

Admittedly, the sentences "John is kissing Mary" and " $2 < 5$ " have sufficient superficial similarity to warrant the same syntactic and semantic analysis. However, these sentences have radically different linguistic environments. For instance, "John is kissing Mary" can be extended to "John is kissing Mary twice", but " $2 < 5$ " cannot be extended to " $2 < 5$ twice". Furthermore, the sentence "John is kissing Mary" is implied by "John is kissing Mary twice". We have therefore to use a logical form that will support this implication. While for 2 and 5 to be related as $2 < 5$ no further entities are required to exist, there have to exist entities beyond John and Mary if "John is kissing Mary (twice)" is to be true, namely at least one (at least two) kiss(es). A plausible formulation is the following:

1. $\exists x(x \text{ is a kiss} \wedge \text{does}(J, x) \wedge \text{suffers}(M, y))$
2. $\exists x, y(x \text{ is a kiss} \wedge x \text{ is a kiss} \wedge \neg x = y \wedge \text{does}(J, x) \wedge \text{it does}(J, y) \wedge \text{suffers}(M, x) \wedge \text{suffers}(M, y))$

Obviously 2. implies 1. in first-order predicate logic. These considerations lead to the distinction between formal and material relations. A relation is *formal* if it holds as soon as its relata are given. Formal relations are called equivalently *immediate relations* since they hold of their relata without mediating additional individuals. The smaller-than relation $<$ for numbers, the instantiation relation, and membership are formal relations. Also the relations denoted by "does" and "suffers" in (1) and (2) are formal. In contradistinction to this, the kissing relation does not hold as soon as its relata exist; a kiss (understood as an individual event) must happen in addition to its relata. Therefore, we call the kissing relation a *material* relation. Formal relations may be classified with respect to the main categories: sets, universals and individuals. *Membership* is a formal relation between sets and between urelements and sets, and *instantiation* is a formal relation between individuals and universals. Universals and sets are connected by the formal relation of *extension*. Examples of formal relations between individuals are *inherence*, *occupation*, *framing*, *containment* which will be explained in section 4.1.

4 Top Level Categories and Upper level Ontologies

In this section further categories of individuals are introduced, and some formal relations are considered in more detail. We call all these categories as ontologically basic because they are the most fundamental.

4.1 Ontologically Basic Categories

The individuals are further subcategorized into *moments*, *substantoid*, *chronoids*, *topoids* and *situoids*¹ -terms which will be explained in more detail in what follows. This yields (at least) the following ontologically basic predicates (also called basic types): $Mom(x)$, $Subst(x)$, $Sit(x)$, $Chron(x)$, $Top(x)$. Chronoids can be understood as temporal durations, and topoids as spatial regions with a certain mereotopological structure. A substance is that which bears individual properties or is connected to other substances by relational moments. Every substance possesses matter. The ultimate (or prime) matter of a substance has no moments or qualities. This implies that an ultimate matter cannot appear since every appearance has to take place via individual moments inhering in the appearing matter. We use the term *substantoid* to cover the whole range between prime matter and those pieces of matter (called substances in the proper sense) which possesses moments including forms, motions and qualities. An *alignment* $\langle a_1, \dots, a_n \rangle$ is a sequence of substantiods a_1, \dots, a_n , the a_i 's are called *components*. An alignment is an individual in contradistinction to a list which is a set.

The origin of the notion of *moment* lies in the theory of “individual accidents” developed by Aristotle in his *Metaphysics* and *Categories* [1]. An accident is an individualized property, event or process which is not a part of the essence of a thing. We use the term “moment” in a more general sense and do not distinguish between essential and inessential moments. Moments include individual qualities, actions and passions, a blush, a handshake, thoughts and so on; moments thus comprehend what are sometimes referred to as “events”. The loss of a moment in this more general sense may change the essence of a thing. Moments have in common that they are all dependent on substantoids. Relational moments are dependent on a plurality of substantoids.

A *situoid* is a part of the world that can be comprehended as a whole and which takes into account the courses and histories of the ontological entities occurring in it. Situations are special types of situoids: they are situoids at a time, so that they present a snap-shot view of parts of the world. Situations can be defined as projections of situoids onto atomic (or very small) time intervals or equivalently as situoids with an atomic (or very small) framing chronoid. An *elementary situation* is composed of individual substances and the relational moments which glue them together. Situations are constituted by using further ontologically basic relations such as causality and intentionality. Analogously, we distinguish elementary situoids from situoids. We assume that every situoid is framed by a chronoid and by a topoid. For every situoid s there is a finite number of universals which are associated with s . $ass(s, u)$ has the meaning: s is a situoid and u is a universal associated with s . There is a basic predicate $Sitel(x)$ for elementary situoid, and a predicate $Sit(x)$ for situoids in general. Our approach to situations differs essentially from that of *Barwise* [2], [3]. Barwise did not elaborate an ontology of relations; thus in particular, the relation of inherence is missing from his theory.

¹ This classification is, of course, tentative.

What are the ontologically basic relations needed to glue together the entities mentioned above? Basic relations are membership denoted by \in , and part-of relations, denoted by $<, \leq$ (reflexive part-of). Other basic relations are denoted by symbols starting with a colon. In the current paper we additionally consider the basic relation of inherence, denoted by $:>$, the relativized ternary part-of relation, symbolized by $:\mu$, the instantiation relation, denoted by $::$, the framing relation, designated by $:\sqsubset$, the containment relation, denoted by $:\triangleright$, and the foundation relation, denoted by $:\perp$.

The phrase “inherence in a subject” can be understood as the translation of the Latin expression *in subjecto esse*, in contradistinction to *de subjecto dici*, which may be translated as “predicated of a subject”. The inherence relation $:>$ - sometimes called ontic predication - glues moments to the substances, which are their bearers, for example it glues your smile to your face, or the charge in this conductor to the conductor itself. The part-of relations $<, \leq$ should have in every case an individual as its second argument, i.e. if $a < b, a \leq b$, then b is an individual. The ternary part-whole relation $:\mu(x, y, z)$ has the meaning: “ z is a universal and x qua instance of z is a part of y ”. Obviously, $:\mu(x, y, z)$ implies $x \leq y$, but not conversely. These universal-dependent part-whole relations are important in applications. The symbol $::$ denotes the instantiation relation, its first argument is an individual, and its second a universal. The instances of a universal u are “individualizations” of the time- and space-independent pattern of features captured by u . The containment relation $:\triangleright$ captures the constituents of a situoid. $x : \triangleright y$ means “ x is a constituent of the situoid y ”. Among the constituents of a situoid s belong the occurring substantoids and the moments inhering in them, but also the universals associated to s .

The binary relation $:\sqsubset$ glues chronoids or topoids to situations. We presume that every situation is framed by a chronoid and a topoid, and that every chronoid or topoid frames a situation. The relation $x : \sqsubset y$ is to be read “the chronoid (topoid) x frames the situation y ”. The defined binary relation $:\perp(x, y)$ expresses the foundation relation between a substance and a moment. $:\perp(x, y)$ has the meaning that x is the foundation of y or x founds the moment y , as for example your hair founds the moment of your *being hairy*, and the moment of *being hairy* inheres in your person. The defined binary relation $:occ(x, y)$ describes a relation between substances and topoids having the meaning: the substantoid x occupies the topoid y .

4.2 Upper Level-Ontologies

We formulate certain necessary criteria which an upper level ontology - in our opinion- should satisfy.

Upper-Level Ontology. Every upper level ontology includes at least the three ontological categories: sets, individuals and universals and a system \mathcal{R} of rela-

tions and predicates containing the basic relations and basic types described in section 4.1.²

We consider the described system of basic categories as a constituent core ontology because individuals are composed of substances and moments or are contained in situoids. This system has to be extended by further basic relations for treating space, time, and shapes. One possibility is to introduce the relation *x is boundary of y*, and the *coincidence relation* [17].

The core ontology should be extended by *unity conditions* and *comprehension principles*. Unity conditions are universals capturing kinds of wholes. An example of a unity condition is the universal *substantoid having a boundary*. Comprehension principles allow to construct new universals from given universals. One method to define comprehension principles is to introduce a language \mathcal{L} whose expressions can be used to formally specify universals. Then, certain expressions can be understood as formal representations of universals. A \mathcal{L} -comprehension principle determines which of the \mathcal{L} -expressions represent (specify) universals.

Using the core ontology (extended by some mereotopological notions) one may introduce following categories: *substrate*, *process*, *event*, *state*, and *continuant*. Here, *substrate* is a universal, and *processes*, *events* and *states* are parts of situoids. A continuant can be understood as a substantoid with boundary.

4.3 Ontology of GOL

In this section we expound the partially axiomatized ontology underlying the modelling language GOL (General Ontological Language). GOL is an ongoing research project at the university of Leipzig [6] which is carried out in collaboration with ontologists at the University of Buffalo [15]. GOL includes the categories which are considered in section 4.1. The modelling language *GOL* is formalized in a first-order language. We presume a basic ontological vocabulary, denoted by **Bas**, containing the following groups of symbols:

Unary basic symbols.

$Ur(x)$ (urelement)	$Set(x)$ (set)	$Ind(x)$ (individual)
$Univ(x)$ (universal)	$Mom(x)$ (moment)	$Subst(x)$ (substantoid)
$Sit(x)$ (situoid)	$Top(x)$ (topoid)	$Chron(x)$ (chronoid)

Symbols for binary and ternary basic relations.

\in (membership)	$::$ (instantiation)	$:>$ (inherence)
$<$ (part-of)	$:\mu(x, y, z)$ (rel. part-of)	$:\sqsubset$ (framing)
$:ass(x, y)$ (<i>y</i> ass. to <i>y</i>)	$:\triangleright$ (is contained in)	$:occ(x, y)$ (<i>x</i> occupies <i>y</i>)
$ext(x, y)$ (is extension)		

² We emphasize that this (neccessary) criterion is not sufficient for an ontology to be upper level.

Furthermore, we use the following symbols: *Sub* (for the universal “substance”), *Time* (for the universal “time”), *Space* (for the universal “space”). To the basic vocabulary we may add further symbols used for domain specific areas; the domain specific vocabulary is called an *ontological signature*, denoted by Σ . An *ontological signature* Σ is determined by a set S of symbols used to denote sets (in particular extensional relations), by a set U of symbols used to denote universals, and by a set K of symbols used to denote individuals. An ontological signature is summarized by a tuple $\Sigma = (S, U, K)$.

The syntax of the language $GOL(\Sigma)$ is defined by the set of all expressions containing the atomic formulas and closed with respect to the application of the logical functors $\vee, \wedge, \rightarrow, \neg, \leftrightarrow$, and the quantifiers \forall, \exists . We use untyped variables x, y, z, \dots ; terms are variables or elements from $U \cup S \cup K$; r, s, s_i, t denote terms. Among the atomic formulas are expressions of the following form: $r = s, t \in s, r \succ \langle s_1, \dots, s_n \rangle, v :: u, t: \sqsubset s, t: \triangleright s$.

The language includes an axiomatization capturing the semantics of the ontologically basic relations. We do not present the axiomatization in full here, but rather illustrate the main groups of axioms by selecting some typical examples. We introduce three groups of axioms, whose union form the axioms $Ax(GOL)$ associated with the language GOL . Besides the logical axioms we have the following groups of axioms.

Axioms of Basic Ontology

(a) Sort and Existence Axioms

1. $\exists x(Set(x)),$
2. $\exists x(Ur(x))$
3. $\forall x(Set(x) \vee Ur(x))$
4. $\neg \exists x(Set(x) \wedge Ur(x))$
5. $\forall x(Ur(x) \leftrightarrow Ind(x) \vee Univ(x))$
6. $\neg \exists x(Ind(x) \wedge Univ(x))$
7. $\forall xy(x \in y \rightarrow Set(y) \wedge (Set(x) \vee Ur(x)))$

(b) Instantiation

D. $ext(x, y) =_{df} Univ(x) \wedge Set(y) \wedge \forall u(u :: x \longleftrightarrow u \in y)$.

1. $\forall xy(x :: y \rightarrow Ind(x) \wedge Univ(y))$
2. $\forall x(Univ(x) \rightarrow \exists y(Set(y) \wedge \forall u(u \in y \longleftrightarrow u :: x)))$

(c) Axioms about sets

1. $\forall uv \exists x(Set(x) \wedge x = \{u, v\})$
2. $\{\phi^{Set} \mid \phi \in ZF\}$, where ϕ^{Set} is the relativization of the formula ϕ to the basic symbol $Set(x)$.³

³ In the formula ϕ^{Set} all variables and quantifiers in ϕ are restricted to sets.

ZF is the system of *Zermelo-Fraenkel*. By 1. and 2. there exists arbitrary finite sets over the urelements. Furthermore, we may construct finite lists. Note, that lists and alignments are different sorts of entities.

(d) Axioms for Moments, Substances, and Inherence

1. $\forall x(Subst(x) \rightarrow \exists y(Mom(y) \wedge y :> x))$
2. $\forall x(Mom(x) \rightarrow \exists y(y \wedge x :> y))$
3. $\forall xyz(Mom(x) \wedge x :> y \wedge x :> z \rightarrow y = z)$

Axiom 3 is called the *non-migration principle*: the substances (considered as an alignment) in which a moment inheres are uniquely determined.

(e) Axioms about Part-of

D1 $ov(x, y) =_{df} \exists z(z \leq x \wedge z \leq y)$, (overlap)

1. $\forall x(\neg x < x)$
2. $\forall xyz(x < y \wedge y < z \rightarrow x < z)$
3. $\forall xy(\forall z(z < x \rightarrow ov(z, y)) \rightarrow x \leq y)$
4. $\forall xyz(:\mu(x, y, z) \rightarrow Univ(z) \wedge x \leq y)$
5. $\forall xyzu(:\mu(x, y, u) \wedge :\mu(y, z, u) \rightarrow :\mu(x, z, u))$

(f) Axioms governing Chronoids and Topoids.

Topoids are three-dimensional spatial regions, chronoids are temporal durations.

1. $\forall x(Sit(x) \rightarrow \exists t_1 t_2(Chron(t_1) \wedge Top(t_2) \wedge t_1 \sqsubseteq x \wedge t_2 \sqsubseteq x))$
2. $\forall x(Chron(x) \rightarrow \exists x_1 \exists s(Chron(x_1) \wedge Sit(s) \wedge x \leq x_1 \wedge x_1 \sqsubseteq s))$
3. $\forall x(Top(x) \rightarrow \exists x_1 \exists s(Top(x_1) \wedge Sit(s) \wedge x \leq x_1 \wedge x_1 \sqsubseteq s))$
4. $\forall xt(occ(x, t) \rightarrow Subst(x) \wedge Top(t) \wedge \forall x(Subst(x) \rightarrow \exists t(occ(x, t)))$

(g) Axioms about situations

D1. $Cont(s) =_{df} \{m \mid m : \triangleright s\}$

D2. $s \sqsubseteq t =_{df} Cont(s) \subseteq Cont(t)$.

1. $\forall x(Mom(x) \rightarrow \exists s(Sit(s) \wedge x : \triangleright s)) \wedge \forall x(Subst(x) \rightarrow \exists s(Sit(s) \wedge x : \triangleright s))$
2. $\forall x(Sit(x) \rightarrow \exists y(Subst(y) \wedge y : \triangleright x))$
3. $\forall xy(Sit(x) \wedge Sit(y) \rightarrow \exists z(Sit(z) \wedge x \sqsubseteq z \wedge y \sqsubseteq z))$
4. $\neg \exists x(Sit(x) \wedge \forall y(Sit(y) \rightarrow y \sqsubseteq x))$
5. $\forall x(Sit(x) \rightarrow \exists y(Univ(y) \wedge ass(x, y))$

Furthermore, we add the following axioms which are related to Σ . $Univ(u)$ for every $u \in U$, $Set(R)$ for every $R \in S$, $Ind(c)$ for every $c \in K$. A knowledge base about a specific domain w.r.t. the signature Σ is determined by a set of formulas from $GOL(\Sigma)$ which are not basic axioms.

5 Comparison to other Upper-level Ontologies

5.1 Knowledge Interchange Format.

Knowledge Interchange Format (KIF) is a formal language for the interchange of knowledge among computer programs, written by different programmers, at different times, in different languages. *KIF* can be considered as a lower-level knowledge modelling language.

The ontological basis of *KIF* can be extracted from [9]; we summarize the main points. The most general ontological entity in *KIF* is an *object*. The notion of an object, used in *KIF*, is quite broad: objects can be concrete (e.g. a specific carbon, Nietzsche, the moon) or abstract (the concepts of justice, the number two); objects can be primitive or composite, and even fictional (e.g. a unicorn). There is no ontological classification of the basic entities. In *KIF*, a fundamental distinction is drawn between *individuals* and *sets*. A set is a collection of objects; an individual is any object that is not a set. *KIF* adopts a version of the Neumann-Bernays-Gödel set theory, *GOL* assumes *ZF* set theory. The functions and relations in *KIF* are introduced as sets of finite lists; here the term “set” corresponds to the term “class”. Obviously, the relations and functions in *KIF* correspond in *GOL* to the *extensional relations*. *KIF* does not provide ontologically basic relations like our inheritance, part-whole and the like. Hence, the ontological basis of *KIF* is much weaker than that of *GOL*. *GOL* can be considered as a proper extension of *KIF*; *KIF* can be understood as the extensional part of *GOL*.

5.2 Upper Level Ontology of Russell and Norvig

The most general categories of this ontology are *Abstract Objects* and *Events* [14]. Abstract objects are divided into *Sets*, *Numbers*, *Representational Objects*; events are classified into *Intervals*, *Places*, *Physical Objects*, and *Processes*. This ontology does not satisfy the criteria of section 4 because there is no clear distinction between sets, universals, and individuals. Also, there is no category of formal relations. The class of universals (here called “categories”) is a subclass of sets. This pure extensional view of universals seems to be not correct. Furthermore, numbers are considered as different from sets. But, it seems to be plausible to introduce numbers as special sets, as in *GOL*. Obviously, classical mathematics can be reduced to set theory. The instantiation relation is identified with membership. Besides this the ontology provides the part-of-relation. The categories of events may be reconstructed within the category of *situoids* in *GOL*. An event in the Russell-Norvig ontology is a “chunk” of a particular universe with both temporal and spatial extent. An interval is an event that includes as subevents all events occurring in a given time period. Such intervals can be, in a sense, understood as *situoids*. But the difference is, that *situoids* are parts of the real world that can be comprehended as a whole. This implies that to *situoids* are associated certain universals capturing the granularity and the view of this part of the world.

5.3 Upper Level Ontology of Sowa

Sowa's ontology [19] does not satisfy the conditions of section 4.2. There is no clear distinction between sets, universals and individuals. A main distinction is made between classes and entities. Since these notions are interpreted in KIF they can be understood within GOL as sets and urelements. There are the following two-place primitive relations: *has*, *instance-of*, *sub-class of*, *temp-part*, *spatial-part*. The instance-of relation is interpreted by the membership-relation, and the ontological status of the *has*-relation is unclear. Since these notions are interpreted in KIF they are considered as set-theoretical notions. We want to emphasize that formal relations of the real world are not relations in the sense of set-theory. Sowa's ontology uses further two epistemic non-KIF operators: *nec* and *poss*; but the ontological character of these modal operators is not clear. In our opinion, modal operators should be eliminated from ontology. In Sowa's ontology several classes are introduced, as for example *relative*, *mediating*, *physical*, *abstract*, *continuant*, *ocurrent*. These notions may be redefined within GOL giving them a clear ontological status.

5.4 Upper Level Ontology of LADSEB

In the papers [12] and [8] some principles for a top-level ontology are outlined. The rudimentary top-level ontology, implicitly given in these papers, partially satisfies our criteria of 4.2. Formal relations are considered in [8] as relations involving entities in all "material spheres"; these have a different meaning from formal relations in our sense which we understand as being "immediate". Then certain formal relations are discussed: instantiation and membership, parthood, connection, location and extension, and dependence. The considered formal properties include concreteness, abstractness, extensionality, unity, plurality, dependence and independence. Instantiation, membership, and parthood are basic relations in our sense; the inherence relation is missing. The extension relation $E(x, y) = x \text{ is the extension of } y$ can be modelled by our relation $occ(x, y)$ (the entity x occupies the topoid y). The *connections relation* from [8] can be defined in GOL by using the coincidence relation and the notion of boundary.

5.5 SUO-Project

SUO is a project recently initiated by the Institute of Electrical and Electronic Engineers to develop a "Standard Upper level Ontology" based on KIF [18]. This is designed to provide definitions for between 1000 and 2000 general purpose terms in such a way as to yield a common structure for low-level domain ontologies of much larger size and more specific scope. The ontologies were first divided into two classes, those defining very high-level concepts and those defining lower-level notions. The highest level contains J. Sowa's upper-level ontology and Russell-Norvig's upper-level ontology, and the second level contains a number of further concepts. From this and the above considerations of sections 5.2 and 5.3 follows that the SUO-ontology does not satisfy our criteria for an upper-level ontology.

6 Conclusions

The development of an axiomatized and well-established upper-level ontology is an important step towards a foundation for the science of Formal Ontology in Information Systems. Every domain-specific ontology must use as a framework some upper-level ontology which describes the most general, domain-independent categories of reality. For this purpose it is important to understand what an upper-level category means, and we proposed some conditions that every upper-level ontology should satisfy. The development of a well-founded upper-level ontology is a difficult task that requires a cooperative effort to make significant progress.

References

1. Aristoteles. Philosophische Schriften, Felix-Meiner Verlag, Hamburg, 1995
2. Barwise, J., J. Perry. Situations and Attitudes; Bradford Books, The MIT Press, 1983
3. Barwise, J. Situations in Logic. CSLI No. 17, 1989
4. Casati, R. A.C. Varzi. The Structure of Spatial Localization; *Philosophical Studies*, 82 (1996), 205-239.
5. Degen, W. : *Some principles of Non-migration* in: Metaphysics in the Post-metaphysical Age, 22nd International Wittgenstein Symposium, Kirchberg am Wechsel 1999, 147-154
6. Degen, W., B. Heller, H. Herre. Contributions to Ontological Foundation of Knowledge Modelling, Report Nr. 02 (2001); Institute für Informatik, Universität
7. Degen, W., B. Heller, H. Herre, B. Smith. GOL: Towards an Axiomatized Upper-Level Ontology, to appear in the Proceedings of the FOIS'2001.
8. Gangemi, A., N. Guarino, C. Masolo, A. Oltramari. Understanding Top-level distinctions, LADSEB-Report (2001)
9. Genesereth, M.R., R. E. Fikes: Knowledge Interchange Format, Version 3.0, Reference Manual, Logic Group Report Logic-92-1, Computer Science Department, Stanford University, Stanford, California
10. <http://www.ontology.uni-leipzig.de>
11. Guarino, N., P. Giaretta: Ontologies and Knowledge Bases (Towards a Terminological Clarification) in N.J.I. Mars (ed.) *Towards Very Large Knowledge Bases*, IOS Press, 1995, Amsterdam
12. Guarino, N. Some Ontological Principles for Designing Upper Level Lexical Resources, First Conference on Language Resources and Evaluation, May 1998
13. F. P. Ramsey. *Universals* in : F. P. Ramsey : Foundations (edited by D. H. Mellor) 1978
14. Russell, S., P. Norvig. Artificial Intelligence, Prentice Hall, 1995
15. <http://philosophy.buffalo.edu/faculty/smith>
16. Smith, B. On Substances, Accidents and Universals. In defence of a constituent Ontology. *Philosophical Papers* 26, (1997), 105-127
17. Smith, B. Boundaries: An Essay in Mereotopology, from L. Hahn, ed. *The Philosophy of R. Chisholm*, LaSalle: OpenCourt, 1997, 534-561
18. SUO: <http://suo.ieee.org/>
19. Sowa, J. <http://www.bestweb.net/sowa/ontology/toplevel.htm>

From Domain Ontologies to Object-Oriented Frameworks

Giancarlo Guizzardi^{1,2}, Ricardo de Almeida Falbo¹, José Gonçalves Pereira Filho¹

Computer Science Department¹
Federal University of Espírito Santo
Fernando Ferrari Avenue,
CEP 29060-900 - Vitória - ES - Brazil
e_mail: {falbo, zegonc}@inf.ufes.br

Centre for Telematics and Information Technology²,
University of Twente
P.O. Box 217, 7500 AE, Enschede,
The Netherlands
guizzard@cs.utwente.nl

Abstract: Ontologies are becoming an important mechanism to build information systems. Nevertheless, there is still no systematic approach to support the design of such systems using tools that are common to information systems developers. In this paper, we propose an approach for deriving object frameworks from domain ontologies and then we show the application of this approach in the software process domain.

Keywords: Ontologies, frameworks, methodologies for using ontologies.

1. Introduction

An Information system cannot be written without a commitment to a model of the relevant world —commitments to entities, properties, and relations in that world. Data structures and procedures implicitly or explicitly make commitments to a domain ontology [1].

Several projects in AI have focused on using ontologies to promote knowledge sharing, and to substitute the usual database or object-oriented schema with an ontology, which offers a semantically richer model of the domain [2]. This trend has also acquired followers in the Software Engineering community. However, one of the major drawbacks to a wider use of ontologies in this area is the lack of approaches to insert ontologies in a more conventional software development process.

Since the current leading paradigm in Software Engineering is the object technology, we claim that we need a systematic approach to derive object models from ontologies in order to put ontologies in practice. In this paper we propose a systematic approach to derive reusable object artifacts from domain ontologies. In section 2, we briefly discuss some aspects of ontology building, including a method and a graphical language, and a past experience using them. In section 3, we present a set-based

formalism for ontology representation. In section 4, we describe our approach to derive object models and frameworks from domain ontologies, showing how it was applied in the software process domain. In section 5, related works are discussed. Finally, in section 6, we report our conclusions.

2. Ontologies

It is impossible to represent the real world, or even a part of it, with all its details. To represent a phenomenon or part of the world, which we call domain, it is necessary to focus on a limited number of concepts that are sufficient and relevant to create an abstraction of the phenomenon at hand. Thus, a central aspect of any modeling activity consists of developing a conceptualization: a set of informal rules that constrain the structure of a piece of reality, which an agent uses to isolate and organize relevant objects and relations [3].

According to Guarino [3], “an ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world”. Based on such definition, an ontology consists of concepts and relations, and their definitions, properties and constraints expressed as axioms. An ontology should not be only a hierarchy of terms, but a fully axiomatized theory about the domain [4].

In [4], we proposed a Graphical Language for Expressing Ontologies (LINGO) and a systematic approach for engineering ontologies. In the knowledge acquisition process, the use of a graphical representation is essential in order to facilitate the communication between requirement engineers and experts. In ontology building, such representation is basically a language representing a meta-ontology. Hence, this language has basic primitives to represent a domain conceptualization. In its simplest form, its notations represent only *concepts* and *relations*. Nevertheless, some types of relations have a strong semantics and, indeed, hide a generic ontology, e.g. the *subtype-of* relation and the several types of *mereological* relations. In such cases, specialized notations have been proposed. In fact, this is the striking feature of LINGO and what makes it different from other graphical representations: any notation beyond the basic notations for concepts and relations aims to incorporate a theory. This way, axioms can be automatically generated. These axioms concern simply the structure of the concepts and are said epistemological axioms.

Concerning the approach for building ontologies, the proposed method basically wraps the following activities: purpose identification and requirement specification, ontology capture and formalization, integrating existing ontologies, and ontology evaluation and documentation [4].

Both language and method have been used in the development of complex information systems in areas such as Software Process, Port Management, Steel Metallurgy, Medicine and Media on Demand Management. Although they have proven to be useful, we identify a great concern from the developers: how to put those ontologies in practice, viz., how ontologies can support actual software development?

In [5], we developed a software process ontology and used it to promote knowledge integration in a Software Engineering Environments (SEE). Since the SEE

was implemented using objects, we have to derive an object model from the domain ontology. This represented a design problem that was informally solved. More recently, other developers have experienced the same problem. Based on these facts, the methodology presented in this paper has been proposed

3. A formalism for Ontology Representation

In session 2, we mentioned our previous experience developing a software process ontology. In this experience, we used first-order logic as the language to specify the axioms of the formal theory. First-order logic is widely known for its expressive power and its ontological neutrality, therefore adding minimal ontological commitments. However, due to the goals of this work, it is very convenient to adopt a formalism that lies in an intermediate abstraction level, between first-order logic and object-orientation. For this purpose, we used a hybrid approach based on pure first-order logic, relational theory, and, predominantly, set theory.

The choice to create a language mainly based on set theory was highly motivated by an important issue: set theory is a complementary *extensional* perspective to the *intentional* nature of first-order logic and, at the same time, a natural option as a conceptual model for reasoning about objects. To clarify this point, the following example is used: let the intention of the concept mortal be "*A mortal is an entity whose life ceases in a point of time*". The logic predicate `mortal(x)` states that `x` is a mortal and, therefore, the characteristics defined by the intention of this concept applies to `x`. It also (implicitly) states that $x \in \text{Mortal}$, i.e. to the set of all the elements of the considered world to which the intention of the concept applies. In an object-oriented perspective, if `x` is an instance of `mortal`, it means that `x` belongs to the `mortal` class, i.e. to the set of all instances of the considered world that share the same properties and the same definition.

Because of these characteristics of set theory, to build a model using the proposed set-based language is a very important step in a systematic translation between the logic and the object worlds. Moreover, the language preserves the expressive power of the first-order logic without adding significant ontological commitments, therefore, being suitable to play the same role in the axiomatization process. Although formal, the language is kept as simple as possible, defining only what is absolutely necessary to accomplish its goals.

Finally, it is essential to explain our decision for defining a new set-based formalism instead of using an existent one, for instance Z [17]? The reasons motivating this choice are both philosophical and practical. From a practical point of view, Z has a complicated mathematical notation that contains some language constructs that are unnecessary for ontology specification, e.g. the primitives for method specification. As a consequence of this, practitioners normally find these notations hard to use, mainly if their rather complex textual syntax is the only vehicle to produce specifications. From a philosophical point of view, the language primitives do not offer all the necessary means to express important ontological distinctions. An example of the latter is the fact that the language considers concept relations and properties as equivalent constructs. This consideration holds true from a mathematical perspective but not from an

ontological one. In other words, we claim that Z's set of primitives is neither sufficient nor necessary for ontological formal representation. For a deep discussion about the ontological distinctions between concept relations, roles and properties please refer to [16].

In the next sub-session, the theoretical foundation for our formalism is briefly presented. It is also discussed how the primitives of this formalism are related to the LINGO building blocks.

3.1 - A theoretical foundation for a Set-based language

Sets are collections of zero or more elements whose members are unique and their order is immaterial. Sets can be finite or infinite. Finite sets with a small number of elements are usually represented by the enumeration of their members. Otherwise, they are represented by formation rules or by the definition of the characteristics and properties that all their members must have in common (intention). In our approach, concepts are defined as sets and, as mentioned before, the statement $x \in \text{Mortal}$ commits x to the concept *Mortal*, both intentionally and extensionally.

Another fundamental building block in the LINGO meta-ontology is the primitive *relation*. This primitive represents a semantic link that exists among a set of (one or more) concepts. In our approach, relations are mapped to the synonymous primitive in set theory. In set theory, a n -ary relation can be defined by the n -tuple $R = (C_1, C_2, \dots, C_n, p(x_1, x_2, \dots, x_n))$, where each C_i represents a different set involved in the relation and $p(x_i)$ is a functional predicate open in n variables that maps each element from the cross-product $C_1 \times C_2 \times \dots \times C_n$ in a *true* or *false* value. In this case, the set R^* (solution set) is the subset of $C_1 \times C_2 \times \dots \times C_n$ whose members e_i all satisfy the predicate $p(e_i)$.

Figure 1 shows an example of a binary relation that links the concepts *Person* and *Organization*. The equivalent description in set theory is $\text{contract} = ((\text{Organization}, \text{Person}, \text{contract}(x,y)))$. From now on, the propositional function $p(x,y)$ will be used as synonym of the n -tuple that defines the relation, assuming that the function is defined in some cross-product $C_1 \times C_2 \times \dots \times C_n$.

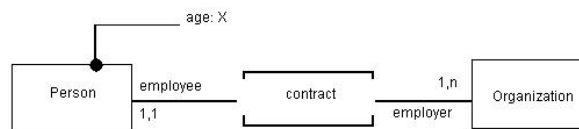


Fig.1 - Example of a binary relation

Figure 1 also depicts two important modeling primitives: *properties* and *roles*. In the ontological level [16] there is a clear distinction between properties, roles and concepts. These distinctions are considered in our approach in a set of mapping directives discussed in section 4. However, at the structural level, the property *age* (of *Person*, fig. 1) represents an ordinary relation $\text{age}(p,x)$ between instances of *Person* and instances of Natural numbers. Actually, this relation happens between the sets

Person and X, where X is a subset of \mathbb{IN} , such that all its members represent the years of existence of another element involved in the age relation $((X \subset \mathbb{IN}) \wedge (\forall x:X, \exists y \text{ age}(x,y)))$. In our formalism, the variable that represents an instance of a property is underlined, e.g. $\forall p:\text{Pessoa}, \exists \underline{a}:X \text{ age}(\underline{a},p)$. Finally, in addition to represent characteristics of concepts, properties can characterize relations as well. This feature is discussed in [19].

In set theory, some essential operations are defined to express the relations between sets (\subset - proper-subset or \subseteq - subset; \cup - Union; \cap - Intersection; \setminus - set difference; \wp - power set), properties of sets ($\#$ - cardinality), restriction on relations (\sim - inverse relation, $;$ - relation composition) and relations between sets and their members (\in - Membership) [18]. In addition to this, we use the basic logic operators (\wedge - conjunction; \vee - disjunction; \oplus - exclusive disjunction; \neg - negation; \rightarrow - conditional; \leftrightarrow - biconditional) and quantifiers (\forall - universal; \exists - existential; $\exists!$ - exists one and only one) to form the core of the formalism employed in this work.

In order to extend this core formalism some additional functions are defined. The most important among them is the function called **Image (Im)**. This function plays a fundamental role in the specification of derivation axioms and solution sets to competency questions [6]. The definition of **Im** is given as follows:

$$\begin{aligned} \text{Im}(_, _): \wp(X) \times (X \leftrightarrow Y) &\rightarrow \wp(Y) \\ \text{Im}(S, R) &= \{x:X, y:Y \mid (x \in S) \wedge ((x,y) \in R^*) \bullet y\} \end{aligned}$$

The following axiom also holds for this function:

$$\forall a, b, R \quad b \in \text{Im}(\{a\}, R) \leftrightarrow a \in \text{Im}(\{b\}, R)$$

Using the relation of figure 2 as an example, a possible valid image set could be: $\text{Im}(\{\text{Org1}\}, \text{contract}) = \{\text{John}, \text{Paul}, \text{Mary}\}$ and, consequently, $\text{Im}(\{\text{John}\}, \text{contract} \sim) = \{\text{Org1}\}$. It is important to notice that **Im** is a distributive function, i.e. $\text{Im}(\{\text{John}, \text{Mary}\}, \text{contract}) = \text{Im}(\{\text{John}\}, \text{contract}) \cup \text{Im}(\{\text{Mary}\}, \text{contract})$.

In figure 2, cardinality constraints are used to specify the number of concept instances that can be involved in a relation. The cardinality (0,n) does not impose any restriction and, for that reason, its not graphically represented. Other cardinality possibilities include (0,1), (1,1) and (1,n). Whenever used, these cardinalities incorporate new axioms to the model. In figure 2, the cardinality (1,1) implies that $\forall p:\text{Pessoa} \# \text{Im}(\{p\}, \text{contract}) = 1$ and cardinality (1,n) implies that $\forall o:\text{Organization} \# \text{Im}(\{o\}, \text{contract}) \geq 1$. Although the examples presented above represent only binary relations, the formalism used in this work is expressive enough to model relations of any arity. Likewise, reflective relations (relations between instances of the same concept) and conditional relations (AND and XOR tight relations) can also be represented [4].

4. From Conceptual Models to Computational Infrastructures – The Impedance Mismatch Problem

The problem of consistently generating computational infrastructures from conceptual models has been known for a long time by the software engineering community as the, so-called, *Impedance Mismatch Problem (IM)* [15]. In the scope of this work, the conceptual models are domain ontologies and the computational infrastructures are object-oriented frameworks. The use of domain ontologies to realize the domain analysis activity in a software engineering process contributes with innumerable advantages [19]. However, the impedance mismatch problem is amplified: instead of performing just one step to translate between two levels of abstraction (conceptual models to computational infrastructures), two steps are necessary. The first step is to translate from an ontological level model (domain axiomatized theory) to an epistemological conceptual model (conceptual view of class diagrams) without losing the explicit representation of knowledge. The second step is the translation between the domain model to its computational concretization - an activity that, in domain engineering terms, is called *domain design*.

Our systematic approach to address this two-level IM problem is composed of a set of directives, design patterns and transformation rules. The directives are used to guide the mapping from the epistemological structures of the domain ontology (concepts, relations, properties and roles) to their counterparts in the object-oriented paradigm. Concepts and relations are naturally mapped to classes and associations in an object model, respectively. Properties of a concept shall be mapped to attributes of the class that is mapping the concept. Although this approach works well in most cases, it is worthwhile to point exceptions that we have found:

- some concepts can be better mapped to attributes of a class in an object model because they do not have a meaningful state in the sense of an object model;
- some concepts should not be mapped to an object model because they were defined only to clarify some aspect of the ontology, but they do not enact a relevant role in an object model;
- relations involving a concept that is mapped to an attribute (or that is not considered in the mapping) should not be mapped to the object model.

Furthermore, the directives consider non-trivial mappings, e.g. n-ary relations, relation properties and conditional relations. At last, they advise the choice between primitives to model a domain entity (Guarino discussion about sortals, temporal neutrality and ontological rigidity is a very good example of this [16]).

Besides the epistemological constructs, ontologies explicitly represent knowledge in a signification level through the use of formal axioms. These axioms can be of two types: *consolidation axioms* and *derivation axioms* [4]. The former aims to impose constraints that must be satisfied for a relation to be consistently established. The latter intends to represent declarative knowledge that is able to derive knowledge from the factual knowledge represented in the ontology. To guarantee the fulfillment of the constraints specified by consolidation axioms, we developed a general *precondition pattern*. Afterwards, this pattern is used to compose another pattern called *whole-part*

pattern [19,20]. This was necessary because ontological whole-part relations are not well mapped to aggregations in an object model, i.e. UML notation for aggregation does not guarantee the constraints imposed by the theory (mereology) underlying the proposed notation [4].

For the sake of necessary brevity, the mapping directives and the consolidation axioms will no longer be discussed in this paper. In a future article, we will describe how the *Set* framework presented in the subsection 4.1 can help to solve most of the issues related to the IM problem. The bottom-up construction of our mereological pattern will be presented in yet another article. The emphasis of this paper is on how this framework and a collection of transformation rules can be applied together to generate consonant JAVA implementations for the derivation axioms.

4.1 - The *Set* Framework

The figure 2 shows a support framework that plays a fundamental role in our ontology-to-objects mapping. This framework implements the mathematical properties described by the theoretical foundation presented in section 3. The methods of the *Set* class are summarized in table 1.

The *Set* class is a generic container that is able to hold extension sets for all kinds of concept instances. To be accessible, each member of a set must have a unique identifier. The *SetElement* interface deals with these requirements, providing an identification mechanism through the *getKey* method. For an instance of any class to be held in a *Set*, it must implement the *SetElement* interface. Consequently, the *Set* class is actually a set of *SetElement* instances. The primary key for these elements is typed as *Object*, which is the top-most class in JAVA hierarchy. This is done in order to give the application classes total freedom regarding implementation decisions.

The framework also defines two other classes: *PersistentSet* and *MemberSet*, both sub-types of *Set*. The former is a set that is able to handle its permanent storage in total transparency from the perspective of the class users. When the *store()* method is invoked in a *PersistentSet*, the class performs the serialization of all its members. The original state of the objects (as well as their relations) can be afterwards restored by the invocation of the *retrieve()* method.

Finally, persistent sets can be used as an interesting alternative to implement databases [19]. Using this paradigm, a database can be seen as a **family** \mathfrak{S} (set of sets), which contains all the sets existing in the application. Since, in this case, each set will be a member of another set, they must also be univocally identifiable. The *MemberSet* is, thus, provided to enable this situation.

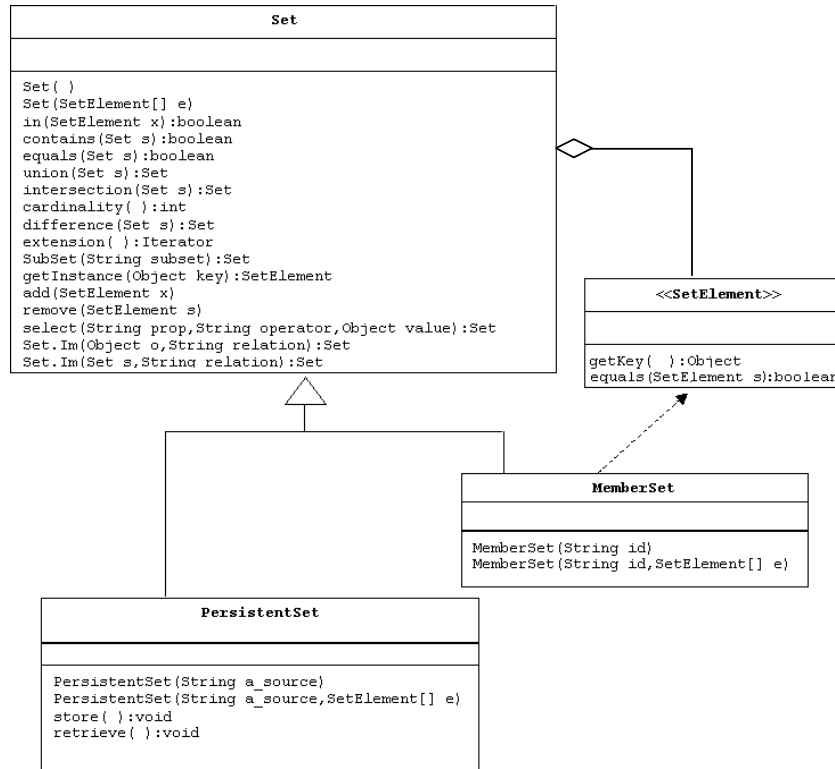


Fig.2 - Framework that implements the mathematical type Set

4.2 – Deriving object frameworks from domain ontologies

Figure 3 shows part of the LINGO model for a *resource-to-activity* allocation theory ontology [5]. The intention is to use it as an example over which we can explain our methodology. Therefore, not only the set of concepts and relations is incomplete and their intentions are not presented, but the fully ontology axiomatization will not be provided as well.

A software engineering process is a composition of a set of inter-related activities. Activities can be either of type construction, management or quality assurance. To make the realization of this process possible, a set of resources is allocated to be used in the activities inside the process. Resources could be of type software, hardware or human resources. Human resources can be managers, developers or project leaders, each one of them with specific skills. Each type of activity requires a special body of knowledge to be performed by a human resource. Thus, for instance, a management activity can only be performed by a Manager. Likewise, a construction activity can only be performed by a Developer. On the other hand, a Project Leader is an experienced human resource that is able to perform any kind of activity.

One central competency question in our example domain is: *for a given human resource allocated to the development of a process, what are the activities he/she can perform?*

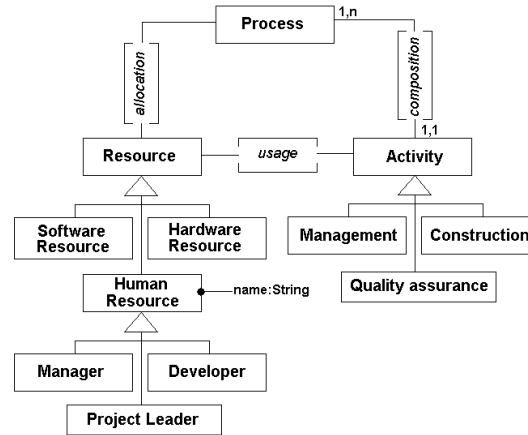


Fig.3 - Simplified ontology for a resource-activity allocation theory

The next subsections present the (partial) axiomatization of this ontology. The following notational convention is used: (C) - concept definition axioms, (R) - Relation definition axioms, (S) - specialization axioms, (CA) - cardinality axioms, (P) - axioms referring to properties and, finally, (O) ontological derivation axioms.

a) Definition Axioms

- | | | |
|----------------------------|---------------------|-----------------------|
| (C1) P = Process | (C2) R = Resource | (C3) A = Activity |
| (C4) H = Human Resources | (C5) M = Management | (C6) C = Construction |
| (C7) Q = Quality Assurance | (C8) D = Developer | |
| (C9) L = Project Leader | (C10) G = Manager | |
- (R1) allocation = (Resource, Process, allocation(r,p))
 (CA1) $\forall a:A \# \text{Im}(\{a\}, \text{composition} \sim) = 1$
 (R2) composition = (Process, Activity, composition(p,a))
 (CA2) $\forall p:P \# \text{Im}(\{p\}, \text{posse}) \geq 1$ (R3) usage = (Resource, Activity, usage(r,a))
 (P1) $\forall h:H \exists ! \underline{n}:\text{String} \text{ nome}(\underline{n},h)$ (S1) M \subset A (S2) C \subset A (S3) Q \subset A
 (S4) H \subset R (S5) T \subset H (S6) D \subset H (S7) G \subset H

As mentioned before, classes define a formation rule for its instance and, therefore, can be seen and manipulated as sets in a meta-level architecture. Consequently, the classification relations in the formalism do not require any specific implementations, i.e. relations such as $a \in A$, are totally resolved by the programming language typing mechanism through the creation of an object a of type A .

Like the classification relation, the sub-type-of relation does not require any additional implementation, i.e. sub-type-of relations among concepts can be directly mapped to generalization/specialization relations among classes.

For the *relation* mapping there are some issues that still must be discussed. In figure 2, a relation contract between the concepts Person and Organization is shown. In our approach, after this relation is translated to an object model, both correspondent

classes have a method `contract`, named after the relation. In this case, the invocation of method `contract()` in an object o_1 of type `Organization`, it is possible to have access to all its hired employees (instances of `Person`). This resulting set is formally specified by the formula $lm(\{o_1\}, contract)$. Likewise, the method invocation in a person instance p_1 returns its contractor, or, $lm(\{p_1\}, contract)$. The returned type of the relation methods depends straightly on the cardinality axioms associated to the relation. For instance, since in the scope of the `contract` relation a `Organization` may have several employees, `contract` is mapped to a `Set` variable in the `Organization` class and, hence, that is the type returned by the invocation of the synonymous method on this class. When a relation has a cardinality axiom imposing an inferior limit equals to 1, this constraint is reflected in the class constructors ensuring the establishment of the relation. As mentioned before, properties can be seen as a special kind of relation between sets and, ergo, the same rules are applied to them.

b) Ontological derivation axioms

(O1) $\forall l:L, a:A \text{ usage}(l,a) \leftrightarrow a \in lm(lm(\{l\}, allocation), composition)$

(O2) $\forall g:G, a:A \text{ usage}(g,a) \leftrightarrow a \in (lm(lm(\{g\}, allocation), composition) \cap M)$

(O3) $\forall d:D, a:A \text{ usage}(d,a) \leftrightarrow a \in (lm(lm(\{d\}, allocation), composition) \cap M)$

The derivation axioms are formalized to answer to the competency questions of the ontology. The axioms above, for instance, answer to the following question: *for a given human resource h_1 , what are the activities he/she could be allocated to?* The solution set for this question must be returned by the invocation of the method `usage()` in an object $h1$ of the `HumanResource` class. However, for this type of methods to be derived from derivation axioms, a set of transformation rules must be defined. The set of rules presented as follows is a subset of our complete set of transformation rules. For a complete set of transformation rules, please refer to [19,20].

T0: $\forall x:X, \forall y:Y \ r_1(x,y) \leftrightarrow y \in C \Rightarrow$

$lm(\{x\}, r_1):Type \equiv C$, such that **if** $\# lm(\{x\}, r_1) = 1$ **then** $Type = Y$ **else** $Type = Set$

This rule states: if for each instance x of type X , x is engaged with all instances y from set C (and only instances of this set) in a relation r_1 , the set returned by the function $lm(x, r_1)$ will be exactly C . The type returned by the method that implements the function in the derived class depends on the cardinality of the relation. Hence, if x is related to only one instance of Y , the returned value shall be of type Y , otherwise, it shall be of type `Set`.

T2: $lm(\{x\}, r_1) \Rightarrow x.r_1()$

As mentioned before, a relation (role or property) r_1 between two concepts X and Y is mapped in the classes that represent these concepts to methods named after the relation. For instance, given an instance x , the invocation $x.r_1()$ returns the set of objects from Y associated to x in the relation r_1 .

T6: $lm(A, r_1) \Rightarrow Set.lm(A, "r_1")$

The rules **T6** promotes the replacement of the mathematical function *Image* by the corresponding syntaxes through which they are implemented in the `Set` class.

T7: $x.r_1():Y \equiv C \Rightarrow$

```
public class X
{
    public Y r1()
    {
        return C;
    }
}
```

Finally, rule **T7** directly translates the axiom written in its left side to the implementation correspondent syntax in the chosen programming language. All the references to the instance *x* existent in the scope of set *C* (to which *x* belongs) are replaced by the JAVA reserved word `this`, so that, references to methods of the same class will be made.

The code fragment below shows the derivation process for the axiom **O1**, and also its implementation in the `ProjectLeader` class.

(O1) $\forall l:L, a:A \text{ usage}(l,a) \leftrightarrow a \in \text{Im}(\text{Im}(\{l\}, \text{allocation}), \text{composition})$	
1. $\text{Im}(l, \text{usage}): \text{Set} \equiv \text{Im}(\text{Im}(\{l\}, \text{allocation}), \text{composition})$	O1, T0
2. $l.\text{usage}(): \text{Set} \equiv \text{Im}(\{l\}.\text{allocation}(), \text{composition})$	1, T2
3. $l.\text{usage}(): \text{Set} \equiv \text{Set}.\text{Im}(\{l\}.\text{allocation}(), \text{"composition"})$	2, T6
4. <code>public class ProjectLeader</code>	3, T7
<pre>{ public Set usage() { return Set.Im(this.allocation(), "composition"); } }</pre>	
<pre>public class ProjectLeader extends HumanResource { ProjectLeader(String name) { super(name); } public Set usage() { return Set.Im(this.allocation(), "composition"); } }</pre>	

5. Related Work

The Peirce project is an international collaborative effort to build a conceptual graph workbench [7][8]. To accomplish interoperability among the different tools produced in the context of the project, a mathematical ontology was proposed and a software library was derived. The ontology contains taxonomic hierarchies for mathematical objects such as: sets, groups, categories, relations, functions, preorders, partial orders and lattices. In [7] a specification for a `Set` class is formalized in several languages (Z, KIF, Conceptual Graphs) and a set of C++ contracts is derived, showing pre/postconditions for the operations of the type. In [7], it is also presented a way to

implement conceptual graphs primitives (such as concepts and relations) in C++ templates. However, due to the focus of these projects the emphasis is on the object-oriented implementation of a CG processor and not in how to create object-oriented artifacts from a conceptual model. Recent research in this field include the Notio Java API [9], always focusing on the object-oriented implementation of the meta-models (CG in this case) and not on the models themselves.

Another very interesting approach to address the impedance mismatch between the ontology and object-oriented abstraction levels is through the use of design patterns. In [12] a set of design patterns for constraint representation in JavaBeans components is presented and computation reflection mechanisms is used to evaluate these constraints at run-time. Likewise, in [13], three design patterns are used promote JAVA implementation for ontologies - represented in the OKBC knowledge model [14]. In this case, ontology concepts are either represented by reflection-backed JavaBeans classes, by an Active Object-Model (AOM), or by a mixed approach based on extending the classes from the AOM.

Constraints are equivalent to, what we call, consolidation axioms. These axioms represent only a subset of the knowledge that must be made explicit in the ontological level. Constraints basically define preconditions that must be satisfied for a relation to be consistently established. As mentioned before, our approach to implement these axioms is also based on ontological design patterns. This approach will be discussed in a future article. In this paper, because of the limited amount of space, we chose to focus on another set of ontological axioms: derivation axioms.

Finally, in [10] and [11], it is presented an approach to create object models such as CORBA IDLs and Java classes and interfaces from Geographic Information Systems (GIS) Ontologies. The papers suggest the automatic generation of interfaces and IDLs from Ontolingua models. These interfaces constitute ontology skeletons that are, afterwards, complemented by implementation code written in Java. Ontology editors, such as Ontolingua, have the ability to create CORBA IDL headers automatically, however in this case, the behavior implementation for the interface methods would still rely on an ad-hoc translation process. Moreover, interfaces alone are not expressive enough to incorporate the knowledge related to all kinds of consolidation axioms, let alone, ontological derivation axioms.

6. Conclusions

Since Aristotle's theory of substance (objects, things and persons) and accidents (qualities, events and process) ontologies have been used in philosophy as a foundation for representing theories and models of reality. Their main purpose is to formally make explicit the semantic distinctions existent in portion of the world, accounted as a domain. Hayes [21] introduced the use of ontologies in Computer Science (more specifically in Artificial Intelligence). Since then, they have been employed in areas such as computational linguistics, knowledge engineering, information integration and multi-agent systems. In addition to that, they have been used in application areas such as enterprise modeling [6] and GIS [11], among several other examples.

In the software engineering realm, domain ontologies have been used to model the foundation over which meta-environments can be constructed [5]. Moreover, they can add important contributions to the domain engineering phase, promoting a reuse-based practice in the requirements engineering level [19].

Nevertheless, few of the ontology construction methodologies lead to executable code and, there was still no systematic approach to fully promote their integration to the object-oriented software development practice. For this reason, most of the object-oriented implementations of domain ontologies rely on informal derivation processes.

In this paper a contribution to address this problem is presented: a methodology through which object-oriented frameworks can be systematically derived from domain ontologies. To accomplish this goal, we also proposed a construction method and a formal representation language. The mathematical foundation of the language (set-theory) highly contributed to the feasibility of our approach. This is mainly due to its suitability to bridge the conceptual and implementation abstraction levels, respectively represented by first-order logic axioms and object models.

The derivation methodology proposed comprises a spectrum of techniques, namely, directives, ontological design patterns and transformation rules. This paper focused on the latter, showing how these rules together with the supporting Set framework can establish a sound path between our formally axiomatized theories and a related consonant implementation in JAVA classes. The other two techniques will be addressed in future articles.

We use the *resource-to-activity* allocation problem in software processes as an example, over which the methodology is presented. The ontology presented was oversimplified due to the lack of space. In despite of that, the methodology has been tested in several case studies, ranging from software process to video on demand management theories. In all these experiences, it was found very effective, mainly because of: (i) its ability to capture the domain knowledge without imposing additional ontological commitments; (ii) its ability to successfully derive object frameworks capable of answering the relevant competency questions.

As it can be seen, our methodology is highly focused on the structural part of domain ontologies, consequently, a natural extension of this work, is to study how a similar approach can be developed to address the dynamics aspects of domains, i.e. behavior ontologies.

References

- [1] Chandrasekaran, B., et al., "*What are Ontologies, and Why Do We Need Them?*", IEEE Intelligent Systems, pp. 20-26, January/February 1999.
- [2] Valente, A., et al., "*Building and (Re)Using an Ontology of Air Campaign Planning*", IEEE Intelligent Systems, pp. 27-36, January/February 1999.
- [3] Guarino, N., "*Understanding, building and using ontologies*", Int. Journal Human-Computer Studies, 46(2/3), February / March 1997.
- [4] Falbo, R.A., et al.; "*A Systematic Approach for Building Ontologies*". Proceedings of the IBERAMIA'98, Lisbon, Portugal, 1998.

- [5] Falbo, R.A., et al.; "*Using Ontologies to Improve Knowledge Integration in Software Engineering Environments*", Proceedings of SCI'98/ISAS'98, Orlando, USA, July, 1998.
- [6] Gruninger, M., and Fox, M.S., "*The Role of Competency Questions in Enterprise Engineering*", *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, Trondheim, Norway, 1994.
- [7] Ellis G; Callaghan S; "*A specification of a Set Class in Peirce*", online: <http://citeseer.nj.nec.com/29926.html>, Oct, 1995.
- [8] Munday C.; Lukose D; "*Object-Oriented Design of Conceptual Graph Processor*", Proceedings of the Fourth International Workshop on Peirce: A Conceptual Graph Workbench University of Maryland, Maryland, USA, 1994.
- [9] Southey, F. and Linders, J. G., "*Notio - A Java API for Conceptual Graphs*", in Proc. of the 7th International Conference on Conceptual Structures (ICCS'99), Springer-Verlag, 1999
- [10] Fonseca, F. Egenhofer M. "*Knowledge Sharing in Geographic Information System*", In: P. Scheuerman, (Ed.) The Third IEEE International Knowledge and Data Engineering Exchange Workshop, Chicago, 1999.
- [11] Fonseca F. T. et al. "*Ontologies and Knowledge Sharing in Urban GIS*", CEUS- Computer, Environment and Urban Systems, 2000.
- [12] Knublauch H.; Sedlmayr M.; Rose T., "*Design Patterns for the Implementation of Constraints on JavaBeans*", NetObjectDays2000, Erfurt, Germany, 2000.
- [13] Knublauch H., "*Three Patterns for the Implementation of Ontologies in Java*", OOPSLA'99 Metadata and Active Object-Model Pattern Mining Workshop, Denver, CO, USA, 1999.
- [14] Grosso W. et al. "*Knowledge Modeling at the Milennium (The Design and Evolution of Protégé-2000)*", Knowledge Aquisition Workshop, Banff, Canada, 1999.
- [15] Woodfield S.N. "*The impedance Mismatch between Conceptual Models and Implementation Environments*", International Conference on Conceptual Modeling (ER'97), Workshop on Behavioral Models and Design Transformations, UCLA, Los Angeles, California, Nov, 1997.
- [16] Guarino N. "*The Ontological Level*". In R. Casati, B. Smith and G. White (eds.), *Philosophy and the Cognitive Sciences*, Vienna, Hölder-Pichler-Tempsky 1994.
- [17] Spivey, J. M. "*Understanding Z: A specification language and its formal semantics*", Cambridge University Press, 1988.
- [18] Roitman J. "*Introduction to modern set theory*", Wiley-Interscience, New York, 1990.
- [19] Guizzardi, G. "*A methodological approach for reuse-oriented software development, based on formal domain ontologies*" (in portuguese), Federal University of Espírito Santo, Master Thesis, 2000.
- [20] Guizzardi, G.; Falbo, R.; Gonçalves J. "*Using Frameworks and Patterns to Implement Domain Ontologies*", Proceedings of the XVI Brazilian Symposium on Software Engineering, Rio de Janeiro, Brazil, Oct, 2001.
- [21] Hayes P. "*The Naive Physics Manifesto*", *Expert Systems in Microelectronics age*, D. Ritchie Ed., Edinburgh University Press, 1978, pp 242-270.

Ontological Aspects in Representing Mathematical Knowledge for Reasoning and Presentation Purposes

Helmut Horacek

Universität des Saarlandes
FB 14 Informatik, Postfach 1150, D-66041 Saarbrücken, Germany
email: horacek@cs.uni-sb.de

Abstract. Workshops on ontologies have been organized frequently over the past decade in connection with AI conferences. This is an indication that the topic is of broad interest to the AI community and also to other research communities, and that pursuing this topic seems to be a long-range term enterprise. One of the most severe problems in the field is the design of ontologies for multiple purposes. We address this issue by discussing the design of the knowledge base of our proof development system Ω MEGA, which combines reasoning facilities for handling mathematical proofs with presentation capabilities in natural language. Supporting the latter requires several enhancements to the primary ontology that is oriented on reasoning purposes. Important extensions are additional conceptual descriptions and markers for roles of defined items. Interfacing reasoning and presentation skills is not only beneficial for illustrating the results of formal inference systems; it is especially valuable for progress in tutorial systems.

1 Introduction

Workshops on ontologies have been organized frequently over the past decade in connection with AI conferences. This is an indication that the topic is of broad interest to the AI community and also to other research communities, and that pursuing this topic seems to be a long-range term enterprise. One of the most severe problems in the field is the design of ontologies for multiple purposes, which in our view, has rarely been addressed explicitly in the literature so far.

We address this issue by discussing the design of the knowledge base of our proof development system Ω MEGA [1], which combines reasoning facilities for handling mathematical proofs with associated presentation capabilities. Supporting the latter requires several enhancements to the primary ontology that is oriented on reasoning purposes. Important extensions are additional conceptual descriptions and markers for roles of defined items, plus an intermediate representation for handling the composition of natural language expressions in a flexible and linguistically motivated way.

This paper is organized as follows. First we characterize requirements on a domain ontology, based on the purposes of the contributing subsystems (here, reasoning and presentation). Then we sketch the present state of our knowledge base from the perspective of its ontology. Thereafter we illustrate extensions that meet the demands of presentation capabilities better than the present ontology does.

2 Task Purposes and Ontological Demands

For our domain of application, high quality representations are required, so that they can only be produced in a hand-crafted manner. Apart from that quality aspect, building ontologies for the domain of mathematics can in some sense be considered less difficult than a similar task for another domains:

- Vagueness does not play a role at all.
- There is a high degree of agreement about the domain concepts; discrepancies merely concern alternative representation variants, formats, and conventions.
- Although the domain as a whole is very large, it can reasonably well be broken down into subdomains of manageable size.

In our application, the underlying ontology (or, better, system of ontologies) has to serve multiple purposes, that is, support domain reasoning and presentations in natural language. Depending on the given task purpose (reasoning or presentation), there exist different demands on the underlying ontology. For purposes of reasoning, the main requirement is an efficient access to definitions and axioms, thereby avoiding redundancy in storing that information to ease maintenance. For meeting this demand, an inheritance network proves to be the best mechanism.

Purposes of presentation impose some additional demands that cannot be met adequately by the inheritance network alone:

- Definitions of items need to be organized in a taxonomy to enable references to categories; in the inheritance network, this information is represented only implicitly. Moreover, items preferred for referring expression need to be marked. For example, the terms 'group' and 'semigroup' are more common than 'monoid', which identifies a similar algebraic structure. The more common terms should be preferred in descriptions even though this may require extended descriptions.
- Conceptual equivalences must be made explicit to avoid redundancy in presentations. There are, for example, various possibilities to define a group in algebraic terms, and switching between definitions might easily result in a strange rephrasing of an obvious equivalence. For reasoning purposes, the equivalence is established by an inference step or, in more complicated cases, by a subproof.
- Additional, highly special conceptual definitions that have no relevance for reasoning purposes may improve the presentation capabilities significantly. Typically, axioms that are expressed as compound or nested rules are good sources for building such definitions, which relate to subexpressions that appear in that axiom and are likely to be described as intermediate results in proofs.

Since reasoning is the primary purpose in the overall system, the knowledge base has been designed to meet the underlying demands best. Presentation capabilities are a typical 'adds-on' feature and not an absolutely required system facility, which constitutes a similar situation as in other systems. Hence, the knowledge base has to be enhanced appropriately to meet presentation demands, too.

3 The Domain Ontology

The enterprise of building an ontology is carried out within the system MBASE [5]. Apart from various measures to make storage and transactions highly efficient, MBASE offers access to a number of mathematical services through a system of mediators that take care of ontological differences between the mathematical service at hand and the view of MBASE (currently supported for three such services).

The statement of a mathematical theorem can depend on the availability of an eventually large set of definitions of mathematical concepts which, in turn, may themselves depend on other concepts. Moreover, previously proven theorems or lemmata may be reused within the context of a proof. Going beyond pure representation purposes, a formal reasoning system needs access to other forms of knowledge, including information about control knowledge for automated reasoners (theorem provers) and about (human-related) presentation knowledge.

In order to store and manipulate these kinds of information, MBASE distinguishes several categories of information objects, on which the structure of the underlying data base model is grounded:

- *Primary (mathematical) objects*, such as symbols, definitions, lemmata, theorems, and proofs.
- *Definitions* for associating meanings to symbols in terms of already defined ones.
- *Assertions*, which are logical sentences, including axioms, theorems, conjectures and lemmata, distinguished according to pragmatic or proof-theoretic status.
- *Human-oriented (technical) knowledge*, such as names for theorems, and specifications for notation and symbol handling.

Based on these categories, the data base model distinguishes the following primary data base objects:

- *Proofs*, as representations of evidence for the truth of assertions.
- *Proof objects*, encapsulating the actual proof objects in various formats, including formal, informal, and even natural language formats.
- *Examples*, due to their importance in mathematical practice.

In addition, the data base model contains some relations between objects of these kinds onto which inheritance is made. These include

- *Definition-entailment*, to relate defined symbols to others and, ultimately, to symbols marked as primitive.
- *Depends-on/Local-in*, which specify dependency and locality information for primary knowledge items. At the present stage of development, this relation is implemented for definitions and proofs, which make explicit the use of symbols/ lemmata in a definition or assertion, as well as for theories, which specifies the organization of mathematical subdomains and the associated inheritance of their properties.

For the task of presentation, most of these objects and relations are relevant, although we do not exploit the full potential at the current stage of development. At present, we make use of human-oriented objects to address technical presentation issues, and we combine definitions, assertions, and proofs along the relations among these objects, selected according to the requirements of a given presentation goal.

4 Methods for Increasing Presentation Capabilities

The techniques described so far mainly support the coordination of static knowledge originating from heterogeneous sources for problem solving purposes. For presentation purposes, there are two fundamental shortcomings of the present representation which severely limit presentation capabilities:

- The connection between mathematical objects and natural language expressions is too simple, since it is restricted to one-to-one correspondences. This is meaningfully applicable to domain terms, but not to some relations and inference rules.
- The dynamic knowledge (e.g., proofs, examples) is represented too coarse-grained and on a superficial level only, which limits variations for presenting it.

These shortcomings are as fundamental as they are deliberate, since the complexity of the design and development tasks for MBASE is high enough. For future extensions, these are good candidates for linking more linguistically oriented tools. Apart from these long-term issues, we intend to address the specific presentation problems with the currently available material, as outlined in section 2 when discussing the additional demands of presentation purposes, by some simpler measures.

In those places where knowledge about certain roles of definitions is required for presentation purposes, this is provided by some sort of marking and a disciplined domain model design. This measure applies to situation where the domain modeling part (e.g., for axioms) contains definitional equivalences (e.g., for two well-established representation versions of mathematical relations), which are connected by mere implications presently. Similarly, common and less common definitions are marked. Moreover, taxonomic links are introduced where needed.

Next, additional conceptual definitions are provided for meeting presentation demands, which may be required to express uses of complex axioms in inferences. For example, the terms 'meat eater' and 'plant eater' may be used in an elegant way to refer to partial results in involved inferencing about consequences of the eating habits of animals (as in the Steamroller problem [6]). For pure reasoning purposes, there is no need to introduce such definitions – they do not even help in representing intermediate result in the inference process more compactly, since the use of these presentation-oriented concepts for producing text requires some abstraction from the referential forms appearing in the intermediate inference results. The central axiom is

$$\forall x (\text{ANIMAL}(x) \rightarrow (\forall y (\text{PLANT}(y) \rightarrow \text{EATS}(x,y)) \vee \forall y \exists z ((\text{PLANT}(z) \wedge \text{ANIMAL}(y) \wedge \text{EATS}(y,z) \wedge (y < x)) \rightarrow \text{EATS}(x,y))))$$

on the basis of which additional definitions are given (as described in detail in [4]):

$$\begin{array}{lll} \text{PLANT-EATER}(x) & ::= & \forall y (\text{PLANT}(y) \rightarrow \text{EATS}(x,y)) & \text{“}x \text{ is a plant eater”} \\ \text{MEAT-EATER}(x) & ::= & \forall y \exists z ((\text{PLANT}(z) \wedge \text{ANIMAL}(y) \wedge \text{EATS}(y,z) \wedge (y < x)) \rightarrow \text{EATS}(x,y)) & \text{“}x \text{ is a meat eater”} \end{array}$$

Finally, in order to increase the flexibility in which uses of mathematical concepts and associated inferences can be presented by natural language text, we intend to make use of bi-directional mappings between natural language and conceptual representations. The mapping operations provide a considerable degree of paraphrasing capabilities. *Mapping schemata* express local correspondences across representation levels. They define equivalences of the information content associated with individual elements of the target representations – natural language and domain model – and corresponding constructs of the intermediate representation, which may consist of a chunk of elements for target representation elements associated with rich semantics. These operations have been originally designed for NL generation, enabling handling multiple languages [2], and their usefulness for building domain models and exploring alternative versions has been demonstrated in [3]. A methodological description and technical details can be found in [2].

5 Expected Benefits

There are several benefits of having a common database for a highly standardized domain such as mathematics:

- Primarily, inference services, such as the capabilities of provers can be strengthened and made better usable.
- Moreover, improved presentation capabilities contribute to the development of didactic tools. We will soon start a project on a dialog-oriented tutorial system that teaches proof skills, which heavily relies on data from MBASE, be it directly or indirectly through the logical reasoning systems incorporated.

References

1. Christoph Benzmüller, Lassaad Cheikhrouhou, Detlef Fehrer, Armin Fiedler, Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Karsten Konrad, Erica Melis, Andreas Meier, Wolf Schaarschmidt, Jörg Siekmann, Volker Sorge: Ω MEGA: Towards a Mathematical Assistant. In Proc. of the 14th International Conference on Automated Deduction (CADE-97), pp. 252-255, Townsville, Australia, 1997.
2. Helmut Horacek: On Expressing Metonymic Relations in Multiple Languages. *Machine Translation* 11, pp. 109-158, 1996.
3. Helmut Horacek: An Approach to Building Domain Models Interactively. In Proc. of *NLDB-2001*, pp. 7-16, Madrid, Spain, 2001.
4. Helmut Horacek: Expressing References to Rules in Proof Presentations. In Proc. of *IJCAR-2001*, short paper session, Siena, Italy, 2001.
5. Michael Kohlhase, Andreas Franke: MBASE: Representing Knowledge and Context for the Integration of Mathematical Software Systems. *Journal Symbolic Computation* 11, pp. 1-37, 2000.
6. Mark Stickel: Schubert's Steamroller Problem: Formulations and Solutions. In *Journal of Automated Reasoning* 2(1), 1986.

Ontologies, Multi-Perspective Modelling and Knowledge Auditing

John Kingston

Artificial Intelligence Applications Institute
Division of Informatics
University of Edinburgh

Abstract. I have extended an existing ontology for knowledge acquisition to support the task of knowledge auditing. This has required two types of change: the ontology has been extended according to the principles of multi-perspective modelling so that it represents knowledge from the viewpoints of “who”, “what”, “where” and “how”; further work may add “when” and “why” perspectives. It also adds some slots to the class of Publications, which are intended to store information for the purpose of knowledge valuation.

1 Introduction

This paper is a position paper describing work being carried out under the EPSRC-funded Advanced Knowledge Technologies project. It describes an attempt to apply the philosophy of multi-perspective modelling [1] to an existing ontology of knowledge resources, in order to extend that ontology in sensible ways. It also touches on further extension of the ontology to permit evaluation of knowledge resources. The overall aim of the work currently being carried out is to support a knowledge audit – that is, a survey of the knowledge that is available in an organisation.

The goal of this work described in this paper has been to produce an ontology that represents all the aspects of multi-perspective modelling; that is, for any knowledge resource, it can represent *what* it is, *who* possesses it, *how* it is used, *where* it can be found, *when* it is needed and *why* it exists (or why it is useful). If knowledge is collected and is indexed according to all aspects of the above ontology, then it should be possible to browse all the people who possess a particular knowledge resource (or part of it); or all the knowledge resources held by a particular person; or all the activities that can be supported by a particular knowledge resource.

However, a good knowledge audit does not just produce a survey report; it must provide estimates of the “goodness” or “value” of knowledge, and it also ought to encode a number of relationships between knowledge of different types, in order to allow browsing of knowledge from different viewpoints or different foci of interest. This paper briefly discusses the principles of multi-perspective modelling and of knowledge valuation, and then describes the ontology that has been developed, with a couple of examples of its use. The examples are drawn from the self-referential domain of AI researchers, their research and their collaborators – specifically, the researchers and industrial collaborators on the AKT project. The “semantic web researchers” ontology published by the KA2 project was used as a starting point.

2 Multi-Perspective Modelling

The thesis of multi-perspective modelling is that for any “knowledge asset” to be represented adequately, it’s necessary to represent a number of different perspectives on its knowledge – and, possibly, to represent the asset at multiple different levels of decomposition. These ideas are based on those of the Zachman framework [2], and are embodied in various knowledge modelling methods, notably the CommonKADS methodology for knowledge engineering [3].

The Zachman framework (also called the Information Systems Architecture framework) has six columns representing *who*, *what*, *how*, *when*, *where* and *why* perspectives on knowledge, and six rows representing different levels of abstraction (see Table 1). Zachman illustrates the different levels of abstraction using examples from design and construction of a building, starting from the “scope” level (which takes a “ballpark” view on the building which is primarily the concern of the architect, and may

represent the gross sizing, shape, and spatial relationships as well as the mutual understanding between the architect and owner), going through the “enterprise” level (primarily the concern of the owner, representing the final building as seen by the owner, and floor plans, based on architect’s drawings) and on through three other levels (the “system” level, the “technology constrained” level and the “detailed representation” level, respectively the concerns of the designer, the builder and the subcontractor) before arriving at the “functioning enterprise” level (in this example, the actual building). Zachman describes this framework as *“a simple, logical structure of descriptive representations for identifying ‘models’ that are the basis for designing the enterprise and for building the enterprise’s systems”* [2].

So the thesis of multi-perspective modelling is that, in order to capture all the important aspects of a knowledge asset, it is necessary to model (or at least, consider modelling) each of the six columns of the Zachman framework.

Table 1. The Zachman framework (from [2])

	Data “what”	Function “how”	Network “where”	People “who”	Time “when”	Motivation “why”
Objectives/ Scope “contextual”	List of things important to the business	List of processes the business performs	List of locations in which the business operates	List of Organizations important to the business	List of Events significant to the business	List of Business goals/ strategies
Enterprise “conceptual”	e.g. Semantic Model	e.g. Business process Model	e.g. Business legacy systems	e.g. Work Flow model	e.g. Master Schedule	e.g. Business Plan
System “logical”	e.g. Logical data model	e.g. Application Architecture	e.g. Distributed Systems Architecture	e.g. Human Interface Architecture	e.g. Processing Structure	e.g. Business Rule Model
Technology constrained “physical”	e.g. Physical data model	e.g. System Design	e.g. System Architecture	e.g. Presentation Architecture	e.g. Control Structure	e.g. Rule design
Detailed representations “out-of-context”	e.g. Data description	e.g. programs	e.g. Network architecture	e.g. Security Architecture	e.g. Timing Description	e.g. Rule Specification
Functioning enterprise	e.g. Data	e.g. Function	e.g. Network	e.g. Organization	e.g. Schedule	e.g. Strategy

3 Valuation of Knowledge

Valuing knowledge is a difficult task, because measuring knowledge is a difficult task, and determining the contribution of knowledge to an activity is also difficult. Some people have taken an approach of valuing products and partial products, calculating the knowledge required to produce each product as a function of the training time and/or salaries required by practitioners, and then generating statistics such as “return on knowledge” (see e.g. [4]). Others (e.g. England’s North Western AI Applications Group) have focused on *knowledge items* instead [5], but concentrated less on the content of knowledge items than on the relationships with other knowledge items – e.g. “you need to know A before you can understand B”. Such a “knowledge map” could be used in conjunction with acquisition times for each knowledge item to determine approximate total acquisition times for certain knowledge items, and thus to calculate an “opportunity cost” for them.

4 Our Approach

I have devised an approach to knowledge auditing which uses an ontology of knowledge-related terms. The aim is to carry out a knowledge audit of AI research and researchers, and so the terms focus on research topics, publication details, and so on. I used the KA2 Knowledge Acquisition Community ontology [6] as a starting point.¹

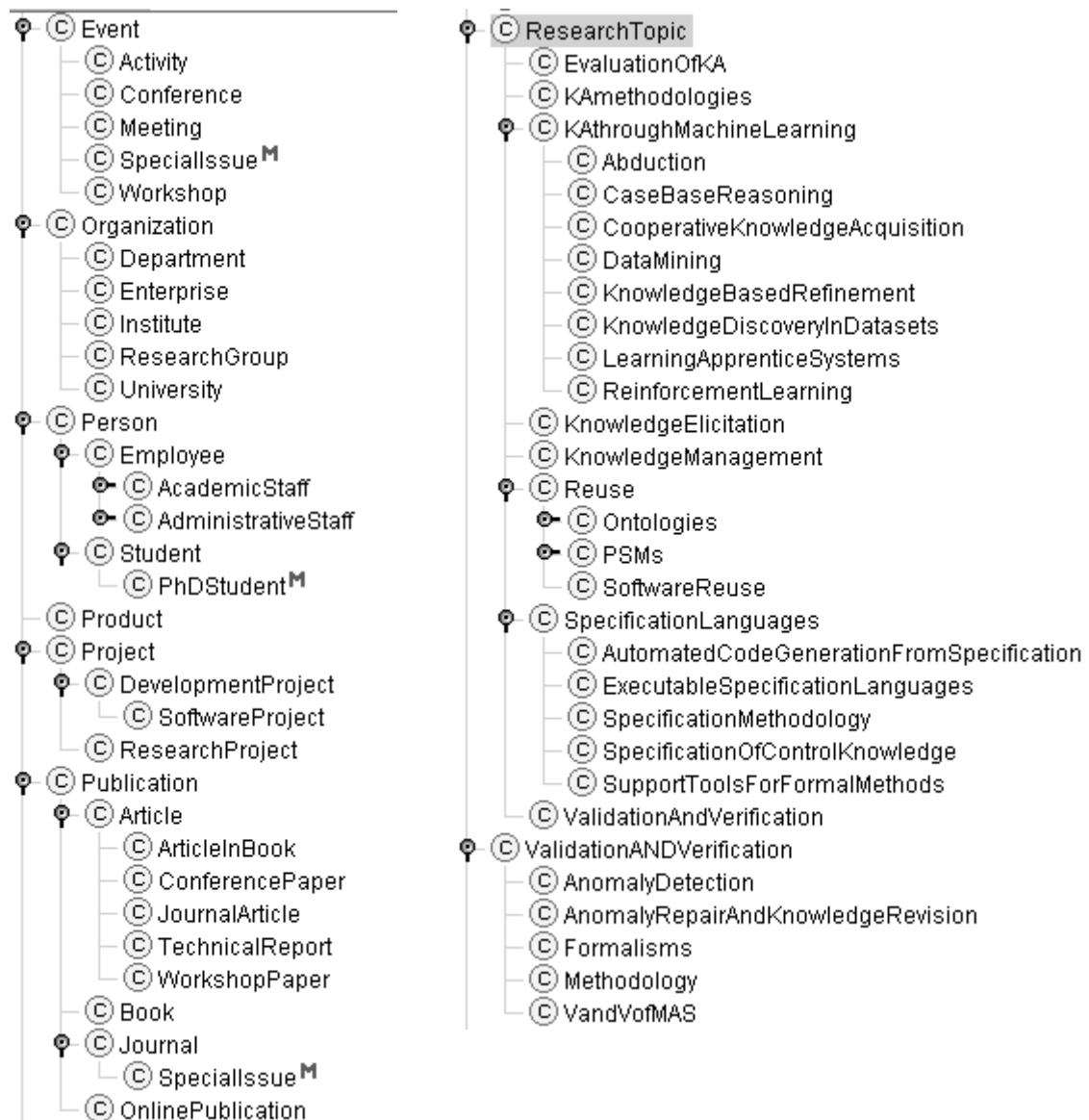


Fig. 1. Top 3 levels of the Knowledge Acquisition Community ontology

The Knowledge Acquisition Community ontology provides the following top level classes: Event, Organisation, Person, Product, Project, Publication, ResearchTopic². I consider that this provides two or three of the six recommended perspectives; the ResearchTopic is the domain (“what”) of a knowledge item, the Person is the owner (“who”), and the location (“where”) of the knowledge is likely to be in a Publication or a Project. I have added an ontology of tasks that need to be performed, which outlines *how* certain top level tasks are carried out; for example, “developing a KBS” breaks down into

¹ Specifically, I downloaded the OIL version, and read it into Protégé-2000 as an RDF Schema file. Some additional work was necessary to recreate slot-to-class links which did not translate correctly.

² ValidationANDVerification is also given as a top level class. I consider this to be poor modelling and have made it a sub-class of ResearchTopic – see Figure 2.

several subtasks, which are themselves decomposed down to the level of tasks such as “detecting circularity in rule based systems”. As yet, there are no specific ontologies of *when* knowledge is created or used (the date of a publication is considered sufficient) nor of *why* knowledge exists or is needed, but such ontologies could be developed if needed – for example, for time-critical applications or applications where storage space or acquisition time is limited.

Using this ontology, it should be possible to encode every relevant piece of information about a knowledge item. For example, the ontology describes the publication details of the paper “COVERAGE: Verifying Multiple-Agent Knowledge-based Systems” [7] covers the research topics of Verification and Validation, Anomaly Detection and Agent Based Systems; the task supported is Anomaly Detection in Multi Agent Systems; and the knowledge owner is the author of the paper (Alun Preece). The paper was published in a journal, whose details are duly recorded, and its publication year was 1999, so it is recent work. An ontology such as this that contained details of all the papers by a large group of researchers should provide the ability to move seamlessly between different views on knowledge; e.g. “show me all the people who know about agent based systems” or “show me all the work on KBS design after 1996”.



Fig. 2. Revised ontology showing “who”, “where”, “what” and “how” ontologies (left) and the current full ontology of tasks (right)

5 Ontology for Knowledge Valuation

However, we not only want to capture all the information about the knowledge present in a group of AI researchers, we also want to attach valuations to this knowledge. The reason is simple; even with a relatively small group of researchers, there may well be a problem of information overload because there is so much knowledge available about certain topics. The aim of knowledge valuation is to identify those items of knowledge that are (or are considered to be) more valuable.

Within the ontology, this is achieved by adding a couple of slots and an entire new ontology section. The slots are attached to the class of Publications; one is intended to record usage statistics (i.e. number of clicks for an online publication, or number of times borrowed for a paper publication), while the other records the “publication quality”, which contains one or more terms from the new section of the ontology. These terms correspond to heuristics which give guidance on the quality of a publication; for example, “Few References”, “Several References” or “Many References”; “One Reference Discussed

In Detail” or “More than One Reference Discussed In Detail”; and on the software side, “Prototype with One Example”, “Proof of Concept system”, or “Fully Functional system”. An unlimited number of these entries can be attached to any publication (or project report), with the intention of being used as guidance on the quality of a publication.

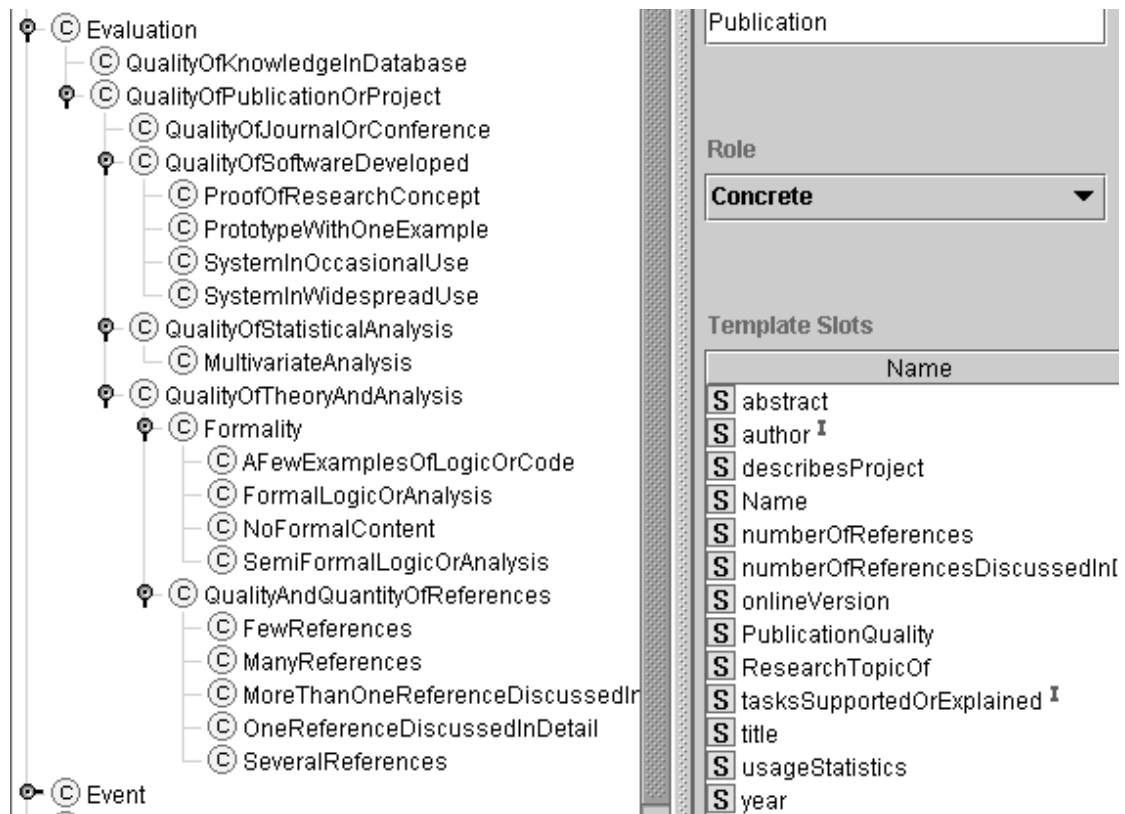


Fig. 3. Ontology for evaluation (left) and slots for Publication class (lower right). Note that several slots exist in order to collect valuation information

6 Future Activities

The priority in future work is to discover related research work and to use that to make further extensions to the ontology, either in the multi-perspective modelling area or in the valuation area.

We also have ongoing work on construction of a knowledge based system to help in determination of the quality of knowledge in databases. This system works by using heuristics to determine the likelihood of obtaining useful knowledge from the database with data mining techniques.

Another piece of work concerns the development of ontology meta-data to allow merging of knowledge models from different sources. This should be coordinated with the multi-perspective ontology work.

Acknowledgements

I would like to acknowledge the efforts of all my co-workers on the AKT project. I would especially like to acknowledge Stuart Aitken for guidance on existing valuation methods and for general advice.

This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

The EPSRC and the Universities comprising the AKT IRC are authorised to reproduce and distribute reprints for their purposes notwithstanding any copyright annotation hereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of the EPSRC or any other member of the AKT IRC.

References

1. J. Kingston and A. Macintosh, "Knowledge Management through Multi-Perspective Modelling: Representing and Distributing Organizational Memory". Research and Development in Expert Systems XVI, proceedings of the Technical stream of ES '99, the 19th BCS SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence, 220-239, 1999.
2. J. Zachman, The Framework for Enterprise Architecture. <http://www.zifa.com/zifajz02.htm>.
3. A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. Van de Velde and B. J. Wielinga. "Knowledge Engineering and Management: The CommonKADS Methodology", MIT Press, ISBN 0262193000. 2000.
4. Knowledge Value-Added (KVA) Methodology Tutorial. <http://www.iec.org/tutorials/kva/>
5. Structural Knowledge Auditing <http://www.nwaiag.com/what/struct.htm>
6. KA2 - Knowledge Acquisition Community Ontology. <http://ontobroker.semanticweb.org/ontos/ka2.html>
7. A. Preece, "COVERAGE: Verifying Multiple-Agent Knowledge-based Systems", Knowledge Based Systems, 12, 37-44, 1999.

The Ontology of Commonsense Knowledge

Walid S. Saba

Knowledge and Language Engineering Group
School of Computer Science, University of Windsor, Windsor, ON N9B-3P4 Canada
`saba@cs.uwindsor.ca`

Abstract. It is by now widely accepted that a number of tasks in natural language understanding (NLU) require the storage of and reasoning with a vast amount of background (commonsense) knowledge. While several efforts have been made to build such ontologies, a consensus on a scientific methodology for ontological design is yet to emerge. In this paper we suggest an approach to building a commonsense ontology for language understanding using language itself as a design guide. The idea is rooted in Frege's conception of compositional semantics and is related to the idea of type inferences in strongly-typed, polymorphic programming languages. The method proposed seems to (i) resolve the problem of multiple inheritance; (ii) suggest an explanation for polysemy and metaphor; and (iii) provide a step towards establishing a systematic approach to ontological design.

1 Introduction

Recent work in natural language understanding (NLU) seems to be slowly embracing what we like to call the 'understanding as reasoning' paradigm, as it is quite clear by now that understanding natural language is, for the most part, a commonsense reasoning process at the pragmatic level, for example in such tasks as reference resolution, plan recognition, lexical disambiguation, prepositional phrase attachments, temporal coherence, and the resolution of quantifier scope ambiguities. For instance, consider the resolution of 'He' in the following:

John shot a policeman. He immediately
a) *fled away.*
b) *fell down.* (1)

It is quite difficult to imagine how children effortlessly resolve such references, if not by recourse to the commonsense facts that, typically, when *shot*(x, y) holds between some x and some y , x is the more likely subject to flee and y is the more likely subject to fall down. Other examples of commonsense reasoning in language understanding involve the resolution of quantifier scope ambiguities. Consider the following:

$$\left\{ \begin{array}{l} \textit{every} \\ \textit{few} \\ \textit{two} \end{array} \right\} \textit{graduate student}(s) \textit{ at MIT submitted a paper to ACL'99} \quad (2)$$

We argue that the plausibility of wide scope *a* (implying a single paper) increases as the number of students involved in the relation decreases. Lacking a syntactic or a semantic explanation, this inference must be a function of our commonsense knowledge of how the ‘submit’ relation between students and ‘papers’ is typically manifested in the real world. Specifically, this inference is based on our commonsense belief that the *submit* relation between a *student* and a *paper* is typically $[1..m]$ -to-1, where m is some small number. Moreover, different individuals seem to have a slightly different value for m , which is consistent with the findings of Kurtzman and MacDonald (1993) that different individuals seem to have different scope preferences in the same textual context.¹

The ‘understanding as reasoning’ paradigm is certainly not entirely new in NLU research. Within the AI community, this paradigm was implicitly embraced by a number of authors (e.g., see Charniak, 1986; Hirst, 1986; Wilks, 1975; Schank, 1982). Unfortunately, however, these approaches were largely based on *ad hoc* algorithms built on top of *informal* knowledge representations. Due to the lack of formality, these procedures were hopelessly unscalable, and scalability was for the most part attempted by pushing the problem from the procedures to the data; which consequently led to the so-called *knowledge bottleneck*. The lack of progress in solving the *knowledge bottleneck* problem generally led AI researchers to either abandon inferential and knowledge-based approaches in favor of more quantitative approaches (e.g., Charniak, 1993), or to focus almost exclusively on the development of large commonsense knowledge bases (e.g., Lenet and Ghua, 1990). Within linguistics and formal semantics, on the other hand, little or no attention was paid to the issue of commonsense reasoning at the pragmatic level. Indeed, the prevailing wisdom (which might be partly due to lack of progress in AI-based NLU) was that a number of NLU tasks require the storage of and reasoning with a vast amount of background knowledge (van Deemter, 1996), an opinion that led some (e.g., Reinhart, 1997) to conclude that pragmatic approaches are ‘highly undecidable’.

In our view both trends were partly misguided. In particular, we hold the view that while language understanding is for the most part a commonsense reasoning process at the pragmatic level, this reasoning process and the underlying knowledge structures that it utilizes must be formalized if we ever hope to build scalable systems. In this light we believe the work on integrating logical and commonsense reasoning in language understanding (e.g., Allen, 1987; Pereira & Pollack, 1991; Zadrozny & Jensen, 1991; Hobbs, 1985; Hobbs *et al.*, 1993; and more recently Asher & Lascarides, 1998; and Saba & Corriveua, 1997) is of paramount importance.

¹ An inferencing strategy that models individual preferences in the resolution of scope ambiguities at the pragmatic level has been suggested in (Saba and Corriveau, 2001).

Much of this work is directed towards formulating commonsense inferencing strategies to resolve a number of ambiguities at the pragmatic level. Although it has been shown (see Saba & Corriveau, 2001) that these inferences do not always require the storage of and reasoning with a vast amount of background knowledge, it is clear that a number of tasks do require such a knowledgebase. Indeed, substantial effort has been made towards building ontologies of commonsense knowledge (e.g., Lenat & Ghua, 1990; Mahesh & Nirenburg, 1995; Sowa, 1995), and a number of promising trends that advocate ontological design based on sound linguistic and logical foundations have started to emerge in recent years (e.g., Guarino & Welty, 2000; Pustejovsky, 2001). However, a systematic and objective approach to ontological design is still lacking. In particular, we believe that an ontology for commonsense knowledge must be *discovered* rather than *invented*, and thus it is not sufficient to establish some principles for ontological design, but that a strategy by which a commonsense ontology might be **systematically** and **objectively** designed must be developed. In this paper we propose such a strategy.

2 Language Use as Guide to Ontological Design

Our basic strategy for designing an ontology of commonsense knowledge is rooted in Frege's conception of Compositionality. According to Frege (see Dummett, 1981, pp. 4-7), the sense of any given sentence is derived from our previous knowledge of the senses of the words that compose it, together with our observation of the way in which they are combined in that sentence. The cornerstone of this paradigm, however, is an observation that has not been fully appreciated regarding the manner in which words are supposed to acquire a sense. In particular, the principle of Compositionality is rooted in the thesis that "our understanding of [those] words consists in our grasp of the way in which they may figure in sentences in general, and how, in general, they combine to determine the truth-conditions of those sentences." (Dummett, 1981, pp. 5).

This simple idea forms the basis of our strategy in designing an ontology for commonsense knowledge: what language allows one to say about a concept, tells us a lot about the concept under consideration. In other words, the meanings of words (i.e., the concepts), can be discovered from the manner in which the words are **used** in everyday language. As Bateman (1995) has suggested, *language* is the best-known theory on everyday knowledge. Assuming that language reflects thought, therefore, analyzing patterns of everyday language 'use' should provide useful clues to the structure of commonsense knowledge. As a motivating example, consider the nouns *table* and *elephant*, and the adjectives *smart* and *large*, out of which four syntactically well formed and semantically valid adjective-noun combinations can be made. One of these combinations, namely *smart table*, is typically rejected on pragmatic grounds, as it is at odds with our commonsense view of what tables are². In particular, while it is sensible to say *large elephant* and *large table*, a *table* is not the kind of

² For the moment we are not concerned with metaphor.

object for which *smart* applies. This analysis results in the fragment hierarchy shown in figure in 1 below. Note that this kind of analysis is not much different from the type inferencing process that occurs in strongly typed, polymorphic programming languages³.

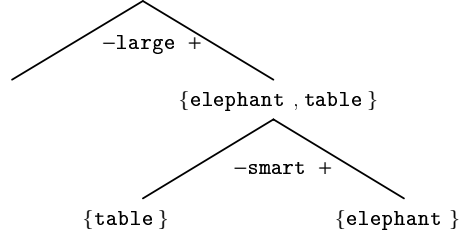


Figure 1. A simple analysis of four adjective-noun combinations.

For example, consider the linguistic patterns and the corresponding type inferences shown in table 1. From $x + 3$, for example, one can infer that x is a **number** since numbers are the "kinds of things" that can be added to 3. In general, the most generic type possible is inferred (i.e., these operations are assumed to be polymorphic).

Linguistic Pattern	Type Inference
$x + 3$	x is number
$\text{reverse}(x)$	x is a sequence
$\text{insert}(x, y)$	x is an object ; y is sequence of x objects
$\text{head}(x)$	x is a sequence
$\text{even}(x)$	x is number

Table 1. Linguistic patterns and the corresponding type inferences.

For example, all that can be inferred from $\text{reverse}(x)$ is that x is the generic type **sequence**, which could be a **list**, a **string** (a sequence of **characters**), a **vector**, etc. Note also that in addition to actions (methods), properties (truth-valued functions) can also be used to infer the type of an object. For example, from $\text{even}(x)$ one can infer that x is a **number**, since lists, sequences, etc. are not the kinds of objects which can be described by the predicate *even*. In general, given a domain (a set of concepts) $C = \{c_1, \dots, c_m\}$ and a set of actions (and properties) $P = \{p_1, \dots, p_n\}$, a predicate $\text{app}(p, c)$ where $c \in C$ and $p \in P$ can be defined such that the action (or property) p applies to (or makes sense of) objects of type c . For each property $p \in P$, a set $C_p = \{c \mid \text{app}(p, c)\}$, denoting all concepts c for which the property p is applicable, can be generated. A concept hierarchy is then systematically discovered by

³ This is reminiscent of (Montague, 1970): "I reject the contention that there is an important theoretical difference between formal and natural languages".

analyzing the subset relationship among the various sets generated. To illustrate how this type of analysis could (systematically) yield a type hierarchy, consider a set of concepts C and a set P of properties (or actions) that may or may not be sensibly applied to concepts in C :

$C = \{\text{list}, \text{string}, \text{set}\}$
 $P = \{\text{empty}, \text{memberOf}, \text{size}, \text{tail}, \text{head}, \text{reverse}, \text{toUpperCase}\}$

Shown in figure 2 below is a number of sets that are generated by $\text{app}(p, c)$ (figure 2a) and the concept hierarchy implied by the subset relationship among these sets (figure 2b). Note that each (unique) set corresponds to a class in the hierarchy. Equal sets (e.g. C_{tail} and C_{head}) correspond to the same class. A class could be given any meaningful label that intuitively represents all the concepts in the class. For example, in figure 2b **sequence** is used to collectively refer to sets, strings and lists.

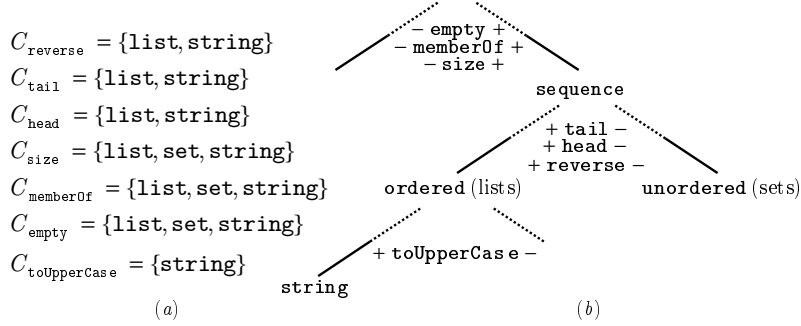


Figure 2. Sets generated by $\text{app}(p, c)$ and the hierarchical structure implied by them.

Clearly, there are a number of rules that can be established from the concept hierarchy shown in figure 2. For example, one can state the following:

$$(\forall c)(\text{app}(\text{reverse}, c) \supset \text{app}(\text{size}, c)) \quad (3)$$

$$(\exists c)(\text{app}(\text{size}, c) \wedge \neg \text{app}(\text{reverse}, c)) \quad (4)$$

$$(\forall c)(\text{app}(\text{tail}, c) \equiv \text{app}(\text{head}, c)) \quad (5)$$

Here (3) states that whenever *it makes sense* to *reverse* an object c , then it also makes sense to ask for the *size* of c . This essentially means that an object to which the *size* operation can be applied must be a parent of an object to which the *reverse* operation can be applied. (4), on the other hand, states that there are objects for which the *size* operation applies, but for which the *reverse* operation does not apply. Finally, (5) states that whenever it makes sense to ask for the *head* of an object then it also makes sense to ask for its *tail*, and vice versa. Thus while there must be at least one property that defines a concept, there could be many (we will have more to say about this below.) Finally, it must be noted that in

performing this analysis we have assumed that the predicate $app(p,c)$ is a Boolean-valued function, which has the consequence that the type hierarchy is a strict binary tree. In fact, this is one of the main characteristics of our method, and has led to two important results: (i) multiple inheritance is completely avoided; and (ii) by not allowing any ambiguity in the interpretation of $app(p,c)$, lexical ambiguity, polysemy and metaphor are explicitly represented in the hierarchy. This will be discussed below.

3 Language and Commonsense Knowledge

The work described here was motivated by the following two assumptions: (i) the process of language understanding is for the most part a commonsense reasoning process at the pragmatic level; and (ii) since children master spoken language at a very young age, children must be performing commonsense reasoning at the pragmatic level, and consequently, they must possess all the commonsense knowledge required to understand spoken language⁴. In other words, we are assuming that deciding on a particular $app(v,c)$ should not be controversial, and that children can easily and consistently answer simple questions such as *do elephants fly*, *do mountains talk*, *do books run*, etc. Note that in answering these questions it is clear that one has to be coconscious of metaphor. For example, while tables, people, and feelings can be strong (i.e., it is quite meaningful to say *strong table*, *strong person*, *strong feeling*), it is clear that the senses of *strong* in these three cases are quite distinct. In fact, the various metaphorical derivations of a lexeme are eventually discovered by the process we describe here, as will become evident in the next sections. The point here is that all that matters, initially, is to consider posing queries such as $app(smart,elephant)$ to a five-year old. Furthermore, in asking such a query we are not asking whether or not every elephant is smart, nor how smart elephants can be, but whether or not it is meaningful to say ‘smart elephant’. We believe that such queries are binary-valued. In other words, while at the quantitative (or a data-level) it could be a matter of degree as to how smart a specific elephant might be, for example, the qualitative question of whether or not it is meaningful to say ‘smart elephant’ is not a matter of degree⁵. With this in mind, our

⁴ It may very well be the case that “everything we know we learned in kindergarten”!

⁵ We will not dwell on this issue too much here except to say that as Elkan (1993) has convincingly argued, to avoid certain contradictions logical reasoning must at some level collapse to a binary logic. While Elkan’s argument seemed to be susceptible to some criticism (e.g., Dubois *et al.* (1994)), there are more convincing arguments supporting the same result. For example, consider the following:

- (1) *John likes every famous actress*
- (2) *Liz is a famous actress*
- (3) *John likes Liz*

Clearly, (1) and (2) should entail (3), regardless of how famous Liz actually is. Using

basic approach to discovering the ontology of commonsense knowledge can be summarized as follows:

- Select a set of adjectives and verbs, $V = \{v_1, \dots, v_m\}$.
- Select a set of nouns $C = \{c_1, \dots, c_n\}$.
- Generate sets $C_i = \{c \in C \mid app(v_i, c)\}$, $1 \leq i \leq m$ for every $v_i \in V$
- Analyse the subset relationship between all sets $C_i \in \{C_1, \dots, C_m\}$

As an initial example, consider the set of verbs $V = \{\text{move, walk, run, talk, reason}\}$ and the set of nouns $C = \{\text{Rational, Bird, Elephant, Shark, Animal, Ameba}\}$. By repeatedly applying $app(v, c)$ the following sets are generated:

```

Cmove = {Rational, Animal, Bird, Elephant, Shark, Ameba}
Ctalk = {Rational}
Creason = {Rational}
Cthink = {Animal}
Cwalk = {Rational, Bird, Elephant}
Crun = {Rational, Bird, Elephant}

```

First we note that while some decisions could ‘technically’ be questioned (say by a biologist), our strategy was to simply consider the question from the point of view of commonsense. In deciding on a particular $app(v, c)$ we considered the query poised to a five-year old: do elephants fly, do they run, do they talk, etc. Questionable situations were simply ignored. This initial process resulted in the hierarchy shown in figure 3 below. Some of the sets indicating positive left and right attributes are given in figure 4 below. Note that some powerful inferential patterns that can be used in language processing are implicit in the structure shown in figure 3. For example, what does not think does not hurt (L_7), what walks also runs (L_8), anything that lives evolves (L_3 and L_4), etc.

4 Knowledge-Level: Being *Locked* to a Property

Motivating our approach to *discovering* (as opposed to *inventing*) the structure of commonsense knowledge is the idea that common language must reflect commonsense, and thus, differences and similarities between concepts must be reflected in our everyday conversation. Technically, the

any quantitative model (such as fuzzy logic), this intuitive entailment cannot be produced (we leave the details of formulating this in fuzzy logic as an exercise!) The problem here is that at the qualitative level the truth-value of *famous(x)* must collapse to either true or false, since at that level all that matters is whether or not Liz is famous, not how famous she actually is.

claim we are making here can be stated as follows: every concept at the knowledge- (or commonsense-) level must 'own' some unique property, and this must also be linguistically reflected by some verb or adjective. This might be similar to what Fodor (1998, p. 126) meant by "having a concept is being locked to a property." In fact, it seems that this is one way to test the demarcation line between commonsense and domain specific knowledge. In particular, it seems that domain-specific concepts are those concepts that are not uniquely locked to any word in the language.

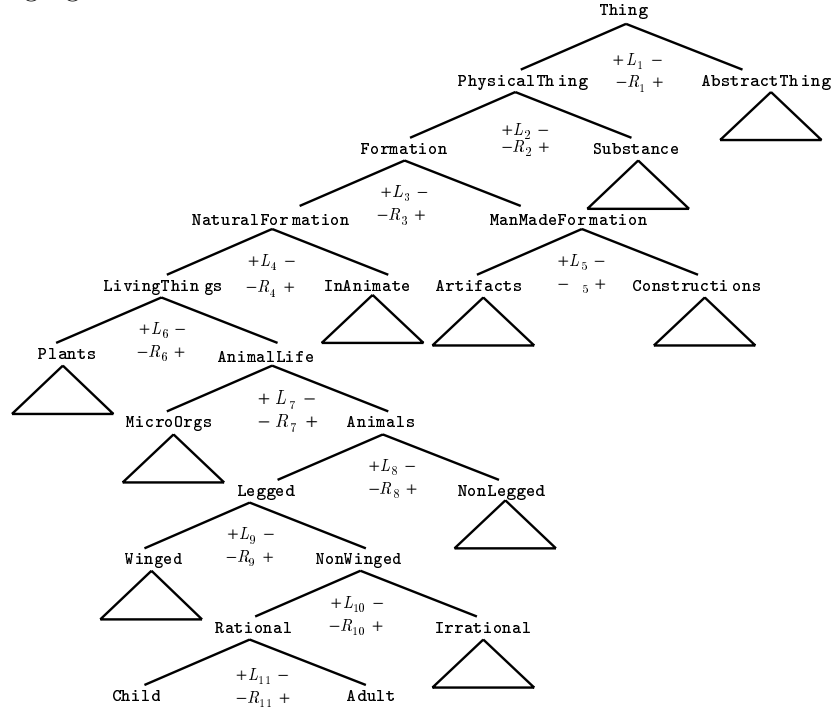


Figure 3. An adult is a physical, living thing that is formed. It evolves, it grows, it develops, moves, it can walk, run, hear, see, talk, think, reason, etc.

$+L_2- = \{develop, form\}$	$+L_3- = \{evolve\}$
$+L_4- = \{live, die, born, grow\}$	$+L_6- = \{branch\}$
$-R_6+ = \{move\}$	$-R_7+ = \{sleep, rest, eat, digest, bleed, hurt, think\}$
$+L_8- = \{sit, jump, walk, run\}$	$-R_9+ = \{roar, cry\}$
$+L_{10}- = \{talk\}$	$-R_{11}+ = \{reason\}$

Figure 4. Sets shown in figure 3.

To illustrate this point further let us analyse a fragment of the hierarchy shown in figure 3 in some detail. The concept `LivingThing`, one can argue, is the concept for which one can say the following:

```
app(LivingThing, Grow)
app(LivingThing, Develop)
app(LivingThing, Live)
app(LivingThing, Die)
```

That is, living things grow, develop, live and die (the concept `LivingThing` is actually 'locked' to several other properties, and thus to several other verbs and adjectives.) What is important to note here is that in order to classify `LivingThing` further one must find some conceptual difference between all living things; a difference which must be somehow reflected in language. As it happens, the word 'move' could be used on all living things except plants. One could, therefore, suggest a classification based on that verb, as shown in figure 3.

Thus all living things are either plant-like things (`Plants`), which are the immobile living organisms, or animal-like things (`AnimalLife`), which are the mobile living organisms. We are now again faced with the same situation, namely further classification of `Plants` and `AnimalLife`. This in turn requires us to find some conceptual difference between potential subtypes of these two concepts, which must translate into a difference in ordinary language. This process is repeated until no further classification based on linguistic differences is possible. In figure 3, for example, `Winged`, which is the class of animals that are legged and have wings was introduced based on the property of *flying*, a difference that translates in language to the lexeme 'fly'. While a biologist can list numerous differences between all birds that fly, classification on such grounds is domain-specific, and does not belong to the knowledge (or commonsense) level, since there does not seem to be a linguistic difference between all birds that fly. This must be, therefore, the knowledge-level, and any further classification beyond this level is based on domain-specific knowledge.

5 Polysemy and Metaphor

In our approach the occurrence of a verb/adjective at any place and at any level in the hierarchy always refers to a *unique sense* of that verb/adjective. Therefore one expects similar senses of a lexeme to apply to concepts along the same path, albeit at different levels in the hierarchy. In particular, one would expect that highly ambiguous verbs to apply to concepts higher-up in the hierarchy, where various similar senses of a verb *v* should end-up applying at various levels below *v*.

Consider for example `form` and `formulate`, in the sense of forming and formulating ideas. Since our method is based on the idea of using such verbs to discover the nature of concepts, `form` and `formulate` must both apply to `ideas`. Note that if everything that can be 'formed' can be 'formulated' and vice versa, then these two verbs would be synonymous.

However, in this case this is not so, since there are things that can be formed but not formulated. For example, consider the small fragment shown in figure 5, where it is shown that 'developing', 'formulating', 'forming', etc. are all specific ways of 'making' (in other words, one sense of 'make' is 'develop'). Note the eventual split however. In particular, while we make, form, and develop both ideas and feelings, ideas are formulated while feelings are fostered.

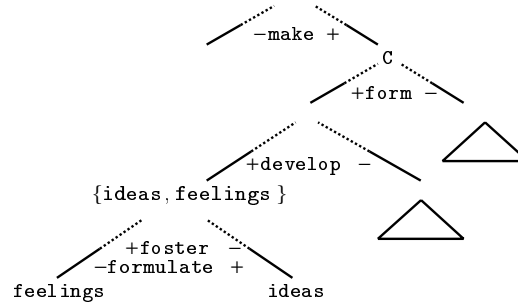


Figure 5. An explanation of polysemy.

While the occurrence of similar senses of verbs at various levels in the hierarchy indicates polysemy, the occurrence of the same verb (the same lexeme) at structurally isomorphic places in the hierarchy indicates metaphorical derivations. Consider the following:

app(run,LeggedThing) (6)
app(run,Machine) (7)
app(run>Show) (8)

(6) through (8) state that we can speak of a legged thing, a machine and a show running. Clearly, however, these examples involve three different senses of the verb *run*. It could be argued that the senses of *run* that are implied by (7) and (8) correspond to a metaphorical derivation of the actual running of natural kinds, the sense implied by (6). It is also interesting to note that these metaphorical derivations occur at various levels: first from natural kinds to artifacts; and then from physical to abstract. Moreover, the mass/count distinction on the physical side seems to have a mirror image of a mass/count on the abstract side. For example, note the following similarity between *water* (physical substance) and *information* (abstract substance):

- *water/information* flows, can be diverted, filtered, processed, etc.
- we can be flooded by, or drown in *water/information*
- a little bit of *water/information* is (still) *water/information*

One interesting aspect of these findings is to further investigate the exact nature of this metaphorical mapping and whether the map is consistent

throughout; that is, whether same-level hierarchies are structurally isomorphic, as the case appears to be so far (see figure 6)⁶.

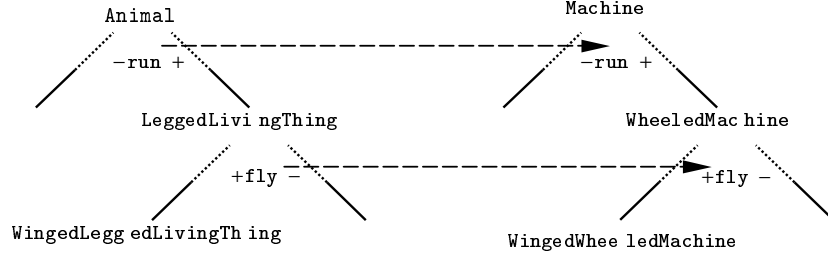


Figure 6. Isomorphic structures explaining metaphors.

6 Negation, Immutable Features and Surprise

The model proposed here allows us to have a very interesting model of the negation. To illustrate, consider the following propositions implied by the concept hierarchy given in figure 3, namely that generally animals move, and people talk:

$$app(\text{Move}, \text{Animal}) \quad (9)$$

$$app(\text{Talk}, \text{Rational}) \quad (10)$$

What is interesting to consider here is how one interprets the negation of such propositions. In particular, there are two possible answers to the query $\neg app(\text{Move}, ?X)$, i.e., to the query "what objects do not move?" One can simply provide $(U - \text{Animal})$ as an answer, where U is the set of all concepts in the universe of discourse. This is the set of all concepts excluding those for which $app(\text{Move}, \text{Animal})$ holds. Thus, plants and all non-living things do not move (see figure 7 below). This is strong negation, since it simply returns the complement with respect to the entire structure. However, we argue that there is a subtle difference between the following queries:

$$\text{Do mountains talk?} \quad (11)$$

$$\text{Do elephants talk?} \quad (12)$$

Although a rational agent would answer "no" in both cases, one might imagine a child replying "nah, mountains do not talk" in response to (11). This must be function of the following: elephants fall directly under the negative polarity of **talk**; while this property is not even applicable to mountains (see figure 7 below). From a Gricean point of view, it seems that "elephants do not talk" is somewhat more meaningful than "mountains do not talk." This subtle difference in the two cases of negation is crucial in performing commonsense reasoning in language

⁶ Conservatively, the mapping might be a homomorphism and not an isomorphism.

understanding. This is also related to the notion of the *immutability* of a feature (Sloman *et al.*, 1998), which is thought to reflect the degree to which a concept depends on a certain feature (or, conversely, how central is a certain feature to the definition of a concept). For example, for some person p , $\neg Run(p)$ is arguably less surprising than $\neg Walk(p)$, which in turn is less surprising than $\neg Move(p)$. That is, it is much harder to assume the immobility of a person than to assume that a certain person might not be able to walk, or run. In our model such measures (including a measure of surprise) could be easily computed.

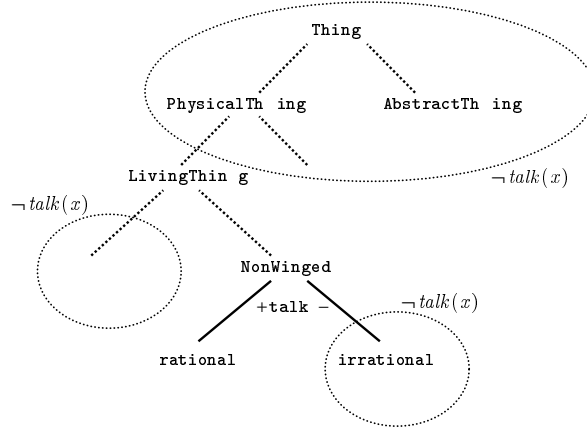


Figure 7. Mountains, elephants, and poems do not talk.

7 Reasoning with Commonsense Knowledge

What we are suggesting in this paper is a process that would hopefully lead to the *discovery* of the ontology of commonsense knowledge. This alone would clearly do little to building natural language understanding systems unless an inferencing strategy that utilizes this ontology is properly formulated. While the ontology provides the synthetic knowledge that an NLU system might need, an NLU system must clearly use quite a bit of analytic knowledge. A typical example would be the following:

$$(\forall p, c_1, c_2)(app(p, c_1) \wedge isa(c_1, c_2) \supset app(p, c_2)) \quad (13)$$

That is, any property that applies to a concept applies to all its subtypes. Clearly there are numerous other such rules that could be added. Another important observation here is that the system that will eventually emerge will yield a much richer type structure than the (flat) type systems typically assumed in formal semantics (e.g., Montague, 1973). For example, from the hierarchy in figure 3 one can clearly establish the following:

$$\text{Walk} : (e_{\text{LeggedThing}} \rightarrow t) \quad (14)$$

$$\text{Write} : (e_{\text{Human}} \rightarrow (e_{\text{Content}} \rightarrow t)) \quad (15)$$

That is, 'write' is not simply a relation between two entities, but a relation between two specific types of entities. Note the importance of this step (of combining formal semantics with a rich type hierarchy), however. For example, (15) states that $\text{write}(x,y)$ is well-typed as long as $\text{isa}(x, \text{Human})$ and $\text{isa}(y, \text{Content})$. In general,

$$\begin{aligned} \text{wellTyped}(\mathbf{v}(e)) &\equiv_{df} \text{type}(\mathbf{v}, (e_m \rightarrow t)) \wedge \text{type}(e, a) \wedge \text{isa}(a, m) \\ \text{wellTyped}(\mathbf{v}(e_1, e_2)) &\equiv_{df} \text{type}(\mathbf{v}, (e_m \rightarrow (e_n \rightarrow t))) \wedge \text{type}(e_1, a) \wedge \text{type}(e_2, b) \\ &\quad \wedge \text{isa}(a, m) \wedge \text{isa}(b, n) \end{aligned}$$

More importantly, however, the combination of a rich type hierarchy and a rigorous semantics should shed some light on the semantics of compound nominals. For instance, type information might explain why removing the middle noun form (16) changes the subject considerably while the same is not true in (17).

$$\text{Computer book sale} \quad (16)$$

$$\text{Information management system} \quad (17)$$

Such rules are important in a variety of language processing tasks, and in particular in topic-based information retrieval. A compositional semantics that exploits a rich type hierarchy should therefore facilitate the development of a meaning algebra; for example to explain why *fake gun* is not (exactly) a gun, whereas *imported gun* is very much a gun. These are precisely the kinds of issues that have prompted this work, and much of this is currently under development.

8 Concluding Remarks

In this paper we argued for and presented a new approach to the systematic design of ontologies of commonsense knowledge. The method is based on the basic assumption that "language use" can guide the classification process. This idea is in turn rooted in Frege's principle of Compositionality and is similar to the idea of type inference in strongly-typed, polymorphic programming languages. The experiment we conducted shows this approach to be quite promising as it seems to have answered a number of questions simultaneously. In particular, the approach seems to (i) completely remove the need for multiple inheritance; (ii) provide a good model for lexical ambiguity and polysemy; and (iii) suggest a plausible explanation of metaphor in natural language.

Much of what we presented here is work in progress, more so than a final result. Therefore, we are well aware that it might be quite ambitious to expect this process to yield a complete classification in a strict binary tree (no multiple inheritance, and no lexical ambiguity). We must also

note that a number of other aspects of this work were not discussed here, such as the part-whole relationship. In particular, it seems that some, but not all, verbs that apply to a concept apply to their parts. For example, *grow* in *app(grow,leg)* and *app(grow,arm)* is very much related to *grow* in *app(grow,person)*. That is, when we refer to a person growing, aging, etc. we are indirectly referring to the growing or aging of the parts. Another important part of this work is to also discover the nature of the relationship between (genuine) types (e.g., **Human**) and roles that concepts play (e.g., **Teacher**, **Father**, etc.) In this regard a number of temporal aspects must also be formalized.

A great deal of work is still needed to formalize the entire approach as well as work out the various inference rules that will eventually be needed in a natural language understanding system. We have already successfully used some of the ideas presented here in NLU tasks, such as developing an efficient and cognitively plausible inferencing strategy to resolve quantifier scope ambiguities at the pragmatic level (see Saba & Corriveau, 2001). While our immediate goal is to discover the ontology of commonsense knowledge, our ultimate goal is to build systems that can *understand* spoken language. This task has proven to be more challenging than has ever been imagined. Turing might have had it right all along: a machine that can converse in spoken language, must be an intelligent machine!

References

- Allen, J. (1987), *Natural Language Understanding*, Benjamin/Cummings, Menlo Park, CA.
- Asher, N. and Lascarides, A. (1998), The Semantics & Pragmatics of Presupposition, *Journal of Semantics*, 15:239-299.
- Bateman, J. A. 1995. On the Relationship Between Ontology Construction and Natural Language: a Socio-Semiotic View. *International Journal of Human-Computer Studies*, 43: 929-944
- Charniak, E. (1993), *Statistical Language Learning*, Cambridge, MA: MIT Press.
- Charniak, E. (1986), A neat theory of marker passing. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 584-588, Los Altos, CA, Morgan Kaufmann.
- Fuzzy logic vs. possibilistic logic, *IEEE Expert*, 9(4), pp. 15-19
- Dummett, M. (1981), *Frege: Philosophy of Language*, Harvard Univ. Press, Cambridge: MA.
- Elkan, C. (1993), The Paradoxical Success of Fuzzy Logic, In *Proceedings of the 11th National Conference on Artificial Intelligence*, AAAI-93, pp. 698-703. AAAI Press.
- Fodor, J. A. (1998), *Concepts - where cognitive science went wrong*, Oxford University Press: NY.
- Guarino, N. and Welty, C. (2000), A Formal Ontology of Properties, In *Proceedings of the 12th Int. Conf. on Knowledge Engineering and Knowledge Management*, LNCS, Verlag.

- Hirst, G. (1986). *Semantic Interpretation and the Resolution of Ambiguity*. Cambridge University Press, Cambridge
- Hobbs, J. (1985), Ontological Promiscuity, In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pp. 61-69, Chicago, Illinois, 1985. ACL.
- Hobbs, J. R., et al. (1993), Interpretation as Abduction, *Artificial Intelligence*, 63:69-142.
- Kurtzman, H. and MacDonald, M. (1993), Resolution of Quantifier Scope Ambiguities, *Cognition*, 48, 243-279.
- Lenat, D. B. and Guha, R.V. (1990), *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley.
- Mahesh, K. and Nirenburg, S. (1995), A Situated Ontology for Practical NLP, In *IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*, IJCAI-95, August 1995, Montreal, Canada.
- Montague, R. (1973). The Proper Treatment of Quantification in Ordinary English, In Thomason (1974).
- Montague, R. (1970), English as a Formal Language, In Thomason (1974).
- Pereira, F. C. N. and Pollack, M. E. (1991), Incremental Interpretation, *Artificial Intelligence*, 50:37-82.
- Pustejovsky, J. (2001), Type Construction and the Logic of Concepts, In P. Bouillon and F. Busa (eds.), *The Syntax of Word Meanings*, Cambridge University Press.
- Reinhart, T. (1997), Quantifier Scope: How Labor is Divided between QR and Choice Functions, *Linguistics and Philosophy*, 20(4): 335-397
- Saba, W. S. and Coriveau, J.-P. (1997), A Pragmatic Treatment of Quantification in Natural Language, In *Proceedings of the 1997 National Conference on Artificial Intelligence (AAAI-97)*, pp. 610-615, Morgan Kaufmann.
- Saba, W. S. and J.-P. Coriveau (2001), Plausible Reasoning and the Resolution of Quantifier Scope Ambiguities, *Studia Logica* (Special Issue on Commonsense Reasoning), 67(2):1-19
- Schank, R. C. (1982), *Dynamic Memory: A Theory of Learning in Computers and People*, Cambridge University Press, New York.
- Sloman, AS., Love, B. and Ahn, W.-K. (1998), Feature Similarity and Conceptual Coherence, *Cognitive Science*, 22(2):189-228.
- Sowa, J. F. (1995). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Boston, MA: PWS Publishing Company.
- Thomason, R. (1974), Editor with an Introduction, *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press.
- van Deemter, K. (1996), Towards a Logic of Ambiguous Expressions, In K. van Deemter and S. Peters (eds.), *Semantic Ambiguity and Underspecification*, pp. 55-76, CSLI, Stanford, CA.
- Wilks, Y. (1975), A Preferential, Pattern-Seeking, Semantics for Natural Language Interface, *Artificial Intelligence*, 6:53-74.
- Zadronzy, W. and Jensen K. (1991), Semantics of Paragraphs, *Computational Linguistics*, 17(2):171-209.

EOS: Making the Epistemic Impact of Ontologies in Knowledge Processing Explicit

Wolfgang Wohner¹

¹Bavarian Research Center for Knowledge Based Systems
Orleansstraße 34, D-81667 Munich, Germany
wohner@forwiss.de

Abstract. EOS provides a formal framework for automated knowledge processing. The EOS framework combines theoretical foundations derived from epistemology and ontology with recent advances in the fields of knowledge representation and processing. As such it allows for an explicit modeling of domain knowledge on the ontological as well as on the epistemic layer, which ensures a flexible, modular system architecture that overcomes the limitations of current systems. We will introduce the overall EOS architecture and discuss its practical applicability by presenting an actual EOS system for intelligent knowledge retrieval in semi-structured data.

1 Introduction

Knowledge processing (KP) has become a major field of interdisciplinary research. Knowledge is being identified, acquired and analyzed, then formalized using various representation models. Resulting representations lay the groundwork for processes that incorporate knowledge, be it on a general level (e.g. for supporting workflows within organizations) or in practical applications (e.g. for assisting specific workflow components). In this paper we will concentrate on the latter use of knowledge, and particularly on machine supported applications, i.e. knowledge *processing* in its strict sense.

At present the predominant means for representing knowledge in KP systems are *formal ontologies*. But despite their widespread use the notion of formal ontologies remains fuzzy. Gruber's commonly accepted definition of an ontology as a '*specification of a conceptualization*' [8] does not account for the semantic implications of ontologies although the object of such conceptualizations is supposed to be knowledge. Consequently, the term 'ontology' is attributed to a large variety of formalizations that differ greatly in their expressive power. What is lacking is an explicit specification of the semantics an ontology is providing and how it may be utilized, i.e. a 'meta-specification' describing the epistemic impact of ontologies. This way KP systems could make immediate use of the semantics inherent in knowledge representations. The benefits of such meta-information on ontologies are:

- On the application side reasoning processes within KP systems do not have to be tailored to particular tasks or domains as ontology-specific inference rules

are provided by the epistemic meta-information. Recognizing epistemics in KP therefore allows for flexible, self-adapting systems. Additionally, exchanging ontologies among different systems is being greatly facilitated by providing their inherent application semantics.

- On the conceptual level epistemic meta-information enriches the expressiveness of knowledge representation languages. The explicit specification of epistemic semantics leads to a clear, two-layered knowledge representation design that respects the distinction between ontological and epistemic facts.

In order to give a deeper conceptual motivation for the EOS framework the following paragraphs of this section will briefly discuss two different perspectives on knowledge, namely organizational knowledge management and the philosophical theory of knowledge, and contrast these approaches with the particular notion of knowledge in machine-supported knowledge processing. Based on these preliminary considerations Section 2 introduces a general framework for knowledge processing systems that serves as a basis for the EOS framework. An actual knowledge processing system complying with the EOS framework is being presented in Section 3 in order to stress its practical applicability. Section 4 comprises related work that is being discussed by referring to the conceptual and practical implications of the EOS framework. Finally, concluding remarks and future research directions can be found in section 5.

1.1 Using Knowledge: The Knowledge Management Business Model

Knowledge management (KM) has become a vital economic factor for organizations as commercial success depends on a proper understanding of internal processes leading to increased productivity and innovation, external processes, e.g. concerning market perspectives, and interactive processes among organizations and their customers as in e-commerce environments. All of these processes require knowledge in order to be mastered successfully. Knowledge, here, is seen in the context of the organizational memory (OM) that comprises the intellectual potential of employees (i.e. skills, experience, expertise), document archives (electronic as well as print media) and all further information relevant to the organization (e.g. inherent in workflow processes) [11]. Therefore, KM tasks concentrate on capturing and organizing OM semantics. Predominant problems in this area are how to implement ways to acquire knowledge, particularly implicit knowledge, and how to incorporate it into the organization's workflows.

The KM business model understands knowledge as a valuable resource that should be exploited in order to supplement the success of an organization. Thus KM offers no explicit theory of knowledge nor does it promote a particular methodology for representing knowledge. Rather, it provides guidelines for identifying and using relevant information (about and within business processes) and its actual and potential benefits for an organization.

1.2 Defining Knowledge: The Philosophical Approach

Theories of knowledge have a long tradition in philosophy, in fact, an entire philosophical discipline, *epistemology*, is dedicated solely to the study of knowledge. Epistemology focuses on questions about the nature of (human) knowledge, i.e. what is knowledge and what can be known. Modern analytical philosophy stresses the propositional structure of knowledge and uses mathematical logic for arguing about propositions. The general idea is that true propositions describe situations in the world, which presupposes objective truth that may be attributed to propositions. In order to turn a true proposition into knowledge its truth has to be proven, or justified. This leads to the most prominent definition of knowledge as ‘justified true belief’ that has been given by the Greek philosopher Plato and is still at the center of current debates.

While epistemology examines the *nature of knowledge* itself, the philosophical discipline of *ontology* is concerned with the *nature of objects* of knowledge. This way ontology serves as a basis for epistemic theories as it gives a notion about being and truth which are fundamental to knowledge.

1.3 Representing Knowledge: The Knowledge Processing Approach

During the past two decades the notion of knowledge in the AI community experienced a notable shift from a primarily functional view that was focussed on modeling human rationality towards a new perspective that put emphasis on modeling systems in the world [1], [9]. We call the former understanding of knowledge the *narrow view* as it is foremost task-driven, i.e. solely knowledge relevant to a specific, pre-defined problem is taken into consideration. Opposed to this notion is the *general view* of the latter approach where knowledge is expected to describe not only details for particular tasks but an entire problem *domain*. Thus the general view is closely related to the objective reality of the problem domain and in itself independent of possible applications. It therefore gives an ontological (in the philosophical sense of the word) perspective on an application area.

Knowledge representations are generally applied to tasks where computer systems need additional input (domain knowledge) in order to adequately process data, e.g. texts in natural language. The system is regarded as possessing knowledge about a problem domain via its formal representation that it accepts as an input. Knowledge in this sense consists of data while *application* logics, i.e. knowledge on how this data may be used, is being considered at most on a restricted level, e.g. languages based on description logics, like DAML+OIL [15], inherently provide a basis for reasoning on the concepts defined but cannot explain any further use of this inferred knowledge within the system, e.g. for query processing. Thus knowledge representations actually address (philosophically) ontological aspects while the definition of epistemic processes that use elements of these representations are to a great extent part of the algorithmic implementation of the system.

2 Processing Knowledge

Knowledge processing (KP) in computer systems comprises all tasks and methods concerned with modeling, representing and employing domain knowledge for enabling a desired system performance, e.g. intelligent information management. In this section we will present a commonly agreed upon design for KP systems and point out its shortcomings. We will use this discussion for motivating a novel, epistemic perspective on KP and present how it may be implemented by the EOS framework.

Example System

In order to motivate these considerations we will first briefly sketch an exemplary KP system that may be used for intelligent information retrieval from heterogeneous information sources such as the Web, interpreted as a vast knowledge base. Representative systems falling into this category are [3] and [12]. The internal KR, typically a formal ontology, models some domain of interest (e.g. the Enterprise Ontology, a collection of terms and definitions relevant to business enterprises [13]) and is used to extract domain specific information from external sources (e.g. Web pages). This information is being stored in the system's own knowledge base and serves as a repository for answering ad-hoc user queries.

2.1 The General Picture

Computer systems that use an explicit modeling of domain knowledge generally exhibit a basic layout as depicted in figure 1. From a functional point of view a KP system will accept input data and process it according to the internal knowledge representation (KR). As indicated the system returns structured information that may on its own part, again, serve as input to the system.

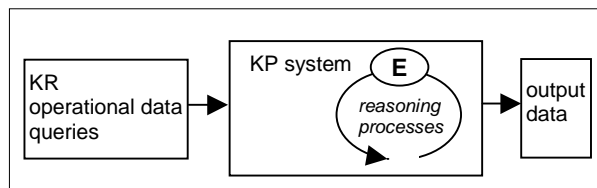


Figure 1. A General Framework for Knowledge Processing Systems

The different system components are:

- **Input data.** Naturally, there are different kinds of input data serving different purposes:
 - **Knowledge representation (KR):** essential to the overall performance is a formalized KR that models the system's application domain. The KR is held persistent within the system and serves as a basis for advanced tasks such as reasoning processes.
 - **Operational data:** any type of documents (e.g. structured or unstructured text files, graphics, etc.) containing information that corresponds to the internal KR is regarded as operational data. For performance reasons it may be stored and indexed separately, e.g. inside a knowledge base attached to the system.
 - **Queries:** user interaction with the system as well as automated processes may trigger queries against the KP system. Queries may address the system's KR itself or be directed towards the structural or semantic content of operational data, resulting in newly created conceptual knowledge that becomes part of the KR, or operational data.
- **Reasoning processes.** The reasoning capacities of the system utilize the internal KR and produce structured information from the previously received input data. Inference semantics and all associated mechanisms for processing an KR instance, i.e. the epistemic layer (E), are integral parts of the system.
- **Output data.** The system returns structured information computed from operational data and/or the internal KR, e.g. specific information held within documents, classifications and indices, or return values of queries.

As depicted by this general layout, the formalization of application semantics (E) is hard-coded into the algorithmical implementation of resulting systems. This leads to several drawbacks:

- KP systems following this general layout must be designed for very specific tasks and domains as the corresponding semantics may differ to a great extent, e.g. natural language processing and deduction on chemical data require very different application semantics.
- Additionally, these systems are highly inflexible regarding conceptual changes of the knowledge representation, i.e. only a restricted class of KRs can be processed by a particular system. Increasing the expressive power of a KR (e.g. by introducing facilities for incorporating axiomatic terms to a given KR model) must therefore result in a costly system redesign.
- For similar reasons exchanging KRs among KP systems poses serious problems. Again, foreign KRs must comply with native application semantics and modeling paradigms of a given KP system in order to render it capable of processing it.

- Finally, even KRs exhibiting the syntactical makeup a KP system can process may be interpreted incorrectly as there is no direct coupling between the objects of the KR and their semantic impact. For example the notion of some relation ‘part-of’ can be different for two KRs, but a KP system will always process ‘part-of’ according to its own implementation.

2.2 The EOS Framework

Based on the preceding considerations we will now introduce the EOS framework for KP systems shown in figure 2. The EOS framework refines and extends the system layout of figure 1 by introducing different semantic layers to the internal knowledge representation.

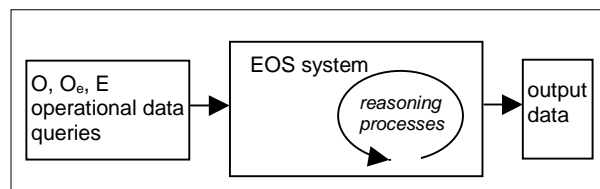


Figure 2. The EOS Framework

We differentiate between two levels of the ontological domain model and an superordinate epistemic layer:

- **Ontological objects of knowledge (O):** a fundamental set of entities the system can identify. This is the ontological basis of the computer system. The general term we use for depicting such ontological entities is that of a *concept*. The basic assumption is that any abstract or concrete real-world entity is being represented by a concept of its own and that no two concepts refer to the same entity (unlike natural language terms that may be used synonymously). This way, concepts allow for a semantically disambiguous modeling of natural objects, their attributes and qualities, as well as relations among objects (e.g. O(business) may contain objects like EMPLOYEE, PROJECT and WORKS-IN, etc.) within some domain of interest.
- **Onto-epistemic objects of knowledge (O_e):** a set of rules or axioms referring to particular concepts of **O**. An example rule of **O**(business) could be paraphrased as ‘each employee works in at least one project’. Onto-epistemic objects complete the ontological domain model by providing domain-specific details to the simple objects of **O**.

- ***Epistemic objects of knowledge (E)***: the set of explications about the objects of **O** and **O_e**. This is the epistemic layer of the system. Unlike with the general framework **E** is here treated as additional input to the system. This establishes an important shift from leaving application semantics hidden within the system to explicitly modeling these semantics into the KR. Explications mould a body of laws that specify how ontological and onto-epistemic objects should be processed by the system. The notion and function of laws will be further elaborated in the following paragraphs.

Formal ontologies in current systems usually cover objects of **O** and to some extent of **O_e** (e.g. [4]). To the best of our knowledge there is no system or methodology using epistemic objects as we understand them, i.e. meta-level descriptions about application semantics of ontology objects. Thus, such a formalization defines metadata about the ontology, foremost semantic processing rules we call *laws* [14]. Again, laws have to be understood and executed by software components but the invaluable benefit they could provide is a homogeneous formal description of the semantic and syntactic implications of such processes. Laws may be regarded as function templates that accept *cases* (e.g. a query) and contain formalized descriptions how to solve them.

Generally, laws provide application semantics about the objects of an KR. As such they render the epistemic impact of an KR explicit while remaining part of the KR (as its metadata). This way the shortcomings of current KP systems complying with the general KP framework can be overcome. The EOS framework allows for flexible, self-adaptable KP systems as knowledge about the semantics of reasoning processes (**E**) is modeled outside these systems (in the form of laws). For example conceptual changes can be expressed by laws which are incorporated into the KR they are describing. Consequently, only the epistemic layer of a KR (i.e. the input data fed to the system) has to be adapted for the system to function correctly. Naturally, processing KRs and passing them over to other EOS systems poses no problems because the application semantics of KR objects is being supplied along with the KR.

Areas of application for laws are:

Ontological semantics

The expressive power of an KR (e.g. if it is possible to define axiomatic terms) is made explicit by laws. Thus they state the representational limits of a KR such as the scope of the ontology or its level of granularity. Therefore an EOS system is aware of representational capacities of a KR it is processing.

Inference semantics

Inference rules may differ greatly between concepts or groups of concepts (e.g. relations ‘is-a’ and ‘part-of’ are both transitive but may be treated differently during query processing). As laws can be general or attributed to single concepts or classes of concepts, they can be used to express inference semantics.

Another aspect of inference semantics concerns *fuzzy concepts* (e.g. closeness) that have to be interpreted according to their context. The meaning of e.g. the term ‘close’ depends on the context of a query or reasoning process, as there are different notions of closeness in the context of houses and, say, atoms. In such cases techniques are needed to establish context which requires laws that describe how the desired information can be deduced.

Query semantics

Automated *semantic query rewriting* is a promising technique for improving query return values. Using ontology knowledge an original query may be transformed into a set of refined queries. The excerpt of an XML document shown below does not contain an <Address> tag, so a query restricted to searching addresses would omit this document:

```
<Person>
<Name> Smith </Name>
<Phone> (222) 333-4444 </Phone>
<Profession> philosopher </Profession>
</Person>
```

By contrast, laws provide rules for extending the scope of the query from addresses to e.g. phone numbers, street names and other address components known to the ontology. This would yield Mr. Smith’s phone number, valuable information that the original query could not have produced.

Uncertainty may also play an important role in the context of iterative document querying, i.e. reasoning on grounds of intermediate results extracted from documents. From the XML example shown above it can be inferred that ‘philosopher’ is an instance of the concept PROFESSION. The value ‘philosopher’ can now be interpreted as a concept as well. But as this information has been derived from the textual content of a document it must be regarded as uncertain knowledge. Uncertain knowledge is an omnipresent factor in intelligent information management and we will intensify our research efforts in that direction.

3 The EOS Knowledge Processing System

Based on the considerations of the preceding sections we will now turn to introducing the architecture of the EOS knowledge processing system, a practical application of the EOS framework. Our EOS system will be used for providing access to heterogeneous semi-structured data sources. Its three main components are (i) an Information Manager, (ii) an Ontology Manager and a (iii) Query Manager.

Basic assumptions about the system are:

- As a preliminary assumption the EOS system possesses a unified interface, i.e. all input and output data is coded using a semi-structured format. In particular, data generated by the system can, again, serve as direct input to the system.
- The system requires a KR that comprises formalized knowledge about the application domain. The ontology fed to the system is expected to contain ontological and onto-epistemic objects of knowledge as well as epistemic laws.
- There is a set of heterogeneous semi-structured documents (e.g. XML documents) covering topics of that domain. These documents serve as operational data of the EOS system.
- There exists a mapping between markup tags of the documents and the concepts of the ontology, i.e. the ontology can ‘understand’ markup semantics in a sense that the concepts involved are part of its formal model.

We decided to concentrate on semi-structured data for several reasons. Besides the most promising perspective that XML-based representation formats will become the predominant means for electronic information exchange and the widespread tool-support for managing and querying XML documents, semi-structured data offers a variety of advantages over unstructured data (e.g. plain text files). The main benefits stem from the distinction between content and metadata which allows for more sophisticated reasoning procedures.

Information Manager

The Information Manager accepts input data (KR and operational data) which is being stored and indexed inside the system’s own internal repository. The stored data must be ready for efficient access, e.g. for reasoning procedures. Operational data will naturally be considerably large, so indexing, along with an efficient linkage between operational data elements and ontology concepts is an essential requirement.

Query Manager

The Query Manager accepts user queries and converts them into queries against the internal repository. Return values can be document fractions, or complete documents, as well as purely ontological data. In order to retrieve valid results from operational data the Query Manager first has to understand the semantics of the query and then make use of the ontology’s domain knowledge for exploring the particular structures of the documents.

Ontology Manager

The domain ontology is being accessed by the Ontology Manager. Its task is to evaluate the epistemic content of ontology laws in order to assist the other system components. Laws are used to control the analysis and processing of objects of **O** and **O_e**. This includes simple ontological reasoning as well as managing more advanced epistemological processes such as semantic query rewriting.

The general task of the EOS system is to derive information (semantics) from semi-structured data (data conforming to syntax). There are some properties of semi-structured data the EOS system may particularly take advantage of. We will illustrate this by referring to XML syntax:

- **Syntax definition:** the syntax definition of markup elements used within an XML document is known via its DTD, so the system is aware of all element names, their attributes and subelements.
- **Concepts:** the semantics of the structuring elements (tags) are known to the system because of the mapping between elements and ontology concepts.
- **Context:** markup elements are organized hierarchically thus establishing contexts (e.g. by nesting tags like <Name> and <Address> into <Person>) which can be interpreted semantically.
- **Types:** in a weak sense each markup element represents a type of its own but it is also possible to introduce primitive or derived element datatypes using e.g. XML Schema.

In summary, semi-structured data offers the possibility to establish a direct linking between the system's knowledge representation and the operational data it has to process. The knowledge representation itself is structured as follows:

- **O:** ontological objects are being defined in a *dictionary of concepts*. On the one hand the dictionary serves as a complete listing of all domain entities the system knows of, along with their definitions in a human readable form. On the other hand it contains a mapping between the concepts and the vocabulary of the problem domain: while ontological concepts are unique and disjunctively refer to single real-world entities the vocabulary used by a community may be ambiguous, i.e. different groups within the same community might also differ in the language they use. Thus, the dictionary provides a mapping for synonymous vocabulary terms depicting the same ontological concept.
A *domain model* formalizes the structure of the problem domain, i.e. its entities and their interrelationships. The domain model is made up of the concepts defined in the dictionary.
- **O_e:** axioms about the concepts of O are specified in a *body of rules*. Following [2] this can be efficiently accomplished using Frame-Logic [10] but DAML+OIL [15] may prove itself feasible as well.
- **E:** application semantics are modeled using *laws*. It is still an open question how laws should be formalized for practical purposes but Frame-Logic and various description logic languages offer a promising starting point for further research.

In conclusion, EOS uses formal ontologies coupled with their inherent application semantics in order to implement intelligent information retrieval and management of semi-structured data. EOS makes use of the notion of laws for operating on ontological information, e.g. for supporting automated reasoning processes and intelligent query processing.

4 Related Work

The discussion of related work will focus on conceptual aspects concerning the formalization of epistemic processes and actual KP systems dealing with semi-structured data.

Modeling Epistemic Processes

A recent approach to bridge the gap between formal ontologies and reasoning processes has been that of Problem-Solving Methods (PSMs). PSMs describe the reasoning process of a knowledge based system in an implementation- and domain-independent way [7]. They are abstract, task-oriented methods designed for facilitating knowledge-engineering processes and address as such similar problems as laws. But, unlike PSMs, laws are an integral part of the formalization of domain knowledge and may be distributed and applied as such. Laws only exist in the context of ontologies and give instructions on how to process their formal objects correctly, i.e. laws are metadata about ontologies while PSMs are about specific (types of) tasks. For knowledge processing purposes we promote the use of laws as they integrate naturally into the formal body of ontologies and KP systems. Consequently, the immediate application of laws poses no difficulty whereas finding an appropriate PSM for a domain-specific task (i.e. choosing a PSM that suits the problem and operates on the correct level of generality) is not trivial. However, PSMs can be useful for designing laws tailored to a particular domain model.

Knowledge Processing on Semi-structured Data

There are two general approaches to combine ontologies and markup languages: (i) defining new markup which is directly related to the ontology or (ii) translating foreign markup into native concepts of the local ontology. The first approach has been propagated by SHOE [12] and Ontobroker [3] but its drawback is obvious. Since their markup methods did not evolve to become widely accepted standards, only a small portion of Web documents use them. For this reason current research besides EOS, e.g. [4], [5], is focused on making the second approach work. Systems such as On2broker [4] use ontologies as the overall structuring principle that drives query processing and inference mechanisms. However, these dynamic processes are mostly hidden within algorithms of software components like query and inference engines. Despite the modular architecture of On2broker (e.g. decoupling of inference and query engines) a notion similar to that of laws is lacking.

5 Conclusions and Future Work

We have motivated and discussed the importance of introducing epistemic metadata to knowledge representations, such as ontologies, used by KP systems. For explanatory purposes we presented a general framework for KP systems and extended

this framework to meet this requirement. The resulting EOS framework exhibits a clear conceptual distinction between ontological, onto-epistemic and epistemic representational objects. We characterized epistemic objects as laws that model application semantics for objects of knowledge on the subordinate ontological and onto-epistemic levels. Finally, we described the architecture of a practical EOS system for intelligent information retrieval. Its main characteristics are, besides the employment of laws, the special notion of concepts as unique representatives of real-world entities and the proposed mapping between these ontological concepts and markup tags of the problem domain.

Our future research will concentrate on a formal specification of laws, and their implementation into our EOS system. This way we hope to establish a sound methodology for the application of laws which will enable us to give out practical guidelines for incorporating them into KP systems. Another interesting aspect of the EOS system is that of semantic query rewriting. We will intensify our efforts in that field and evaluate the capabilities and shortcomings of XML query languages (e.g. XQuery) in that respect.

6 References

1. W. J. Clancey. The Knowledge Level Reinterpreted: Modelling Socio-Technical Systems. *International Journal of Intelligent Systems*, 8: 33-49, 1993.
2. S. Decker, D. Fensel, M. Erdmann, R. Studer. The Technical Core of Ontobroker. Draft, <http://citeseer.nj.nec.com/149833.html>.
3. D. Fensel, S. Decker, M. Erdmann, R. Studer. Ontobroker: The Very High Idea. In *Proceedings of the 11 th International Flairs Conference (FLAIRS-98)*, Sanibel Island, Florida, USA, pp. 131 -135, May 1998.
4. D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, A. Witt. On2broker: Semantic Access to Information Sources at the WWW. In *Proceedings of IJCAI-99 Workshop on Intelligent Information Integration*, Stockholm, 31 July 1999.
5. D. Fensel et al. OIL in a nutshell In: *Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, R. Dieng et al. (eds.), *Lecture Notes in Artificial Intelligence, LNAI*, Springer-Verlag, October 2000.
6. A. Farquhar, R. Fikes, & J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. *Knowledge Systems Laboratory*, September, 1996.
7. A. Gomez-Perez, V.R. Benjamins. Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. In *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, Stockholm, Sweden, August 2, 1999.
8. T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220, 1993.
9. N. Guarino, *Formal Ontology, Conceptual Analysis and Knowledge*

Representation. International Journal of Human and Computer Studies, special issue on The Role of Formal Ontology in the Information Technology edited by N. Guarino and R. Poli, vol 43 no. 5/6, 1995.

10. M. Kifer, G. Lausen, J. Wu: Logical Foundations of Object-Oriented and Frame-Based Languages, Journal of the ACM, 42, 1995.
11. R. Klemke. Context Framework – an Open Approach to Enhance Organizational Memory Systems with Context Modelling Techniques. Proceedings of the Third Int. Conf. On Practical Aspects of Knowledge Management (PAKM2000), Basel, Switzerland, 30-31 Oct. 2000.
12. Luke, S., Heflin J. SHOE 1.01. Proposed Specification. SHOE Project. February 2000. <http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm>.
13. Mike Uschold, Martin King, Stuart Moralee and Yannis Zorgios (1998) The Enterprise Ontology The Knowledge Engineering Review , Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).
14. W. Wohnner. A Modest Proposal: Reasoning Beyond the Limits of Ontologies. In Proceedings of IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, Washington, August 4-5, 2001.
15. Reference description of the DAML+OIL (March 2001) ontology markup language, <http://www.daml.org/2001/03/reference.html>.

An Ontology for WWW Summarization in Bone Marrow Transplantation (BMT): an Interim Report

Brigitte Endres-Niggemeyer¹, Bernd Hertenstein², Claudia Villiger¹, and Carsten Ziegert²

¹ Fachhochschule Hannover/University of Applied Sciences,
Dept. of Information and Communication, Ricklinger Stadtweg 120
D-30459 Hanover, Germany
{Brigitte.Endres-Niggemeyer, Claudia.Villiger}@ik.fh-hannover.de
<http://summit-bmt.fh-hannover.de/>

² Medizinische Hochschule Hannover/Hanover Medical School
Dept. of Hematology and Oncology,
D-30623 Hanover, Germany
Hertenstein.Bernd@mh-hannover.de
Carsten.Ziegert@ik.fh-hannover.de

Abstract. In an interim report, we describe an ontology for WWW summarization in Bone Marrow Transplantation. It is text-based and qualifies as a grounded ontology. In addition, it is user-centered. For stating medical knowledge, we use first-order logic (FOL) extended with contexts. The ontology is prepared for serving several purposes in a summarization system. It will be stored and managed by an XML database server. Currently, it is still under construction, but developed so far that its features can be demonstrated.

1 Introduction

In this paper, we describe an ontology under construction. Its aim is to support efficient search for documents and summarization from the WWW for Bone Marrow Transplantation (BMT), a specialized and life-critical area of hematology.

We are presenting our BMT ontology in an interim report when about 20% of it is realized. The basic argument for doing so is that we should present the ideas at an early stage: nothing really new will come up by waiting for an implementation rate of some 80%, but only more implementation of the same. We cannot expect to one day report about completed work because in a real-world environment, ontology engineering is a never-ending task, sliding softly from a construction mode to the later update mode.

1.1 Background

To our best knowledge, there is no BMT ontology that might support summarizing. While we can use technical features and in part material from existing thesauri and ontologies, we are far from taking over the bulk of the ontology from external sources. A main content reason is that existing ontologies or thesauri, MeSH¹ (Medical Subject Headings) and UMLS (Unified Medical Language System)² included, do not reach the detail needed for summarizing in BMT. This is easily explained: They contain BMT concepts without specifically focusing on them. Their authors certainly never intended to provide concepts for

¹ The database of Medical Subject Headings (MeSH) is available at <http://www.nlm.nih.gov/mesh/MBrowser.html>.

² The metathesaurus of UMLS contains concepts and classifications from more than 60 vocabularies. The metathesaurus is available at <http://umlsks3.nlm.nih.gov/>.

text knowledge processing at a sentence level. For their aims, the vocabulary can be more general. A second main reason is related to text processing. Our ontology is a concept dictionary that must encompass the knowledge for dealing with concept occurrences in text, where the concepts may occur under a different form, in the most simple case as a synonym. This sort of task-specific knowledge is not found in existing ontologies or thesauri. A third problem resides in the reliability of ontologies and thesauri which have only indirect relations to the data they describe. One may doubt whether they have any empirical grounding in current texts of the domain that satisfies standard demands of corpus linguistics. Since text summarization must pave the way from concrete texts to the formal representation and the processing of their meaning, an empirically constructed ontology that is rooted in texts of its domain is the safer approach.

The situation is illustrated by the MIHMA project [15] knowledge representation for bmt line, a web site that serves as the main reference point to BMT material on the Internet for a diversified audience. The knowledge representation has features of an ontology and it focuses on the BMT domain. It draws upon UMLS and is used for retrieving web resources. It is left to a domain expert to judge their relevance and integrate them into bmt line. Applause for a group that shared our domain cannot conceal the fact that their purposes and their ontology content were different from what we need.

Together with many colleagues, Hahn and Reimer [12] favor the use of linguistic and domain knowledge in text understanding. In practice, however, ontologies are not yet often used for the purpose of summarization. WordNet (concepts of common English) is by far the most popular ontology in use, applied for instance in SUMMARIST [14].

Technically speaking, today's choices include WWW-resident representation techniques, for instance XML-based ones inspired by SHOE [13]. We apply them in a situation where different and earlier approaches compete with newer web-driven ones. We adhere to logic representation formalisms. We state our facts and rules in first-order predicate logic extended by contexts [17] and enable safe [8] and simple conclusions.

1.2 Use of the BMT ontology for summarization

Our ontology is a dense representation of domain knowledge in Bone Marrow Transplantation (BMT) tailored to the needs of domain experts. Most of the factual and linguistic knowledge is incorporated in the ontology. For every domain concept, the ontology also encompasses statements of relevant knowledge about it. From an information retrieval perspective, these statements exclude wrong concept combinations. During summarization, a match with propositions helps to establish the relevance of candidate statements in retrieved texts with respect to the current question. For concept identification in running text, the BMT ontology stores lexical equivalents of concepts and the paraphrases by which they are expressed. Like a classic thesaurus, the ontology also conveys information for human users, in particular for query scenario formulation. In the following it is described in some detail how the ontology supports query formulation for information retrieval, text passage retrieval, and summarization proper.

– Query scenario formulation and query expansion

For question-oriented summarization, we need reasonably precise queries. The better the question, the better chances the system has to come up with a helpful response. In order to have users state their queries in a well-structured fashion that the system can interpret, we provide scenarios, query forms specific for types of questions and situations. Into these forms, users fill concepts of the ontology. The scenarios accept statements and questions. Users can browse the concept, synonyms, hypernyms, hyponyms and description fields of ontology records. We provide definitions and descriptions of concepts assembled from many WWW sources and a few others.

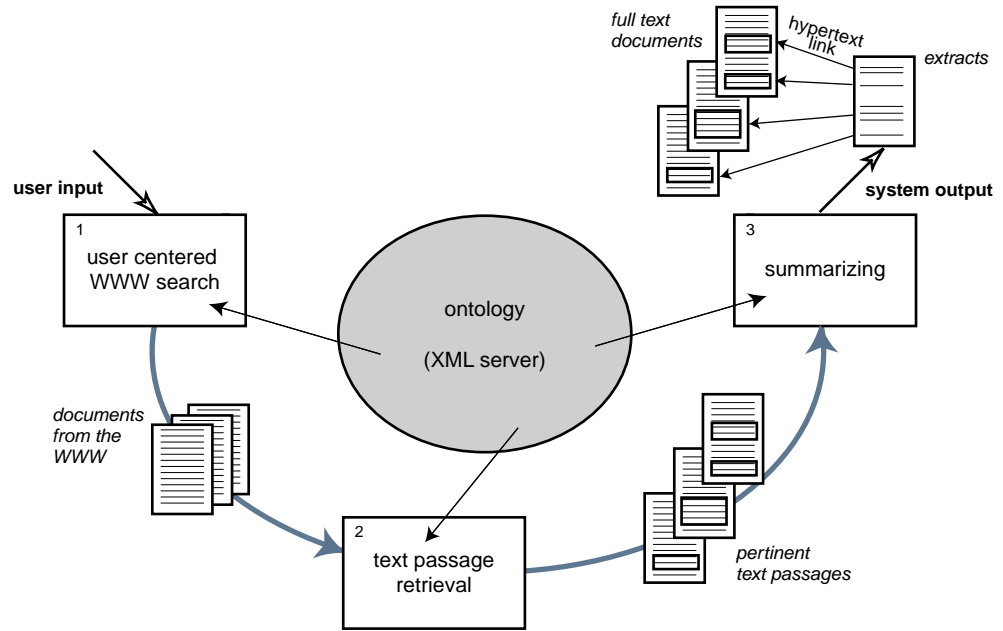


Fig. 1. The ontology as the knowledge server for the system components

After being reformulated in predicate logic form, the query scenario is enriched with related concepts from the ontology, and adapted to the search form of involved search engines (see processing step 1 in figure 1).

- **Text passage retrieval**

As soon as retrieved documents arrive, they are roughly checked for relevance by scanning them for passages that include the query terms. The preliminary relevance check is performed using the concepts of the ontology (see processing step 2 in figure 1).

- **Summarizing**

Summarizing means reducing information to its most important (relevant) points [4]. Most often, the source information is given as a text, and the resulting summary is a short text. Summarization mostly operates at sentence level, and summaries are composed of relevant statements (often sentences), as opposed to passages including unrelated relevant terms as in text passage retrieval. When the summary is to answer a question, only statements that correspond to the question at a sentence level can be relevant, all others are not. So passages that contain ontology concepts used in the query are checked for question-related relevance at sentence level. For this purpose, the ontology must store propositional knowledge with every concept, practically speaking logical propositions that use the current concept as an argument. One of the main differences between thesauri and ontologies is that ontologies state propositional knowledge about their concepts, whereas thesauri are restricted to core paradigmatic relations such as generic ones.

Our system under development uses summarization agents that apply the strategies of human summarizers [4]. They are modelled in terms of cognitive agents (see step 3 in figure 1).

In this paper, we focus on content engineering problems of our ontology. First we describe the empirical method applied for concept acquisition and the organization of the ontology

(see section 2.1). We proceed with a detailed discussion of the formalization process (see section 2.2). After that, we give some technical implementation details.

2 The BMT Ontology for Summarization

2.1 Empirical Ontology Development

In line with others [2, 9] we promote an approach to knowledge modeling based on texts. We argue that papers published in core journals of the domain are safer than other knowledge sources, especially oral information sources such as interviews or expert meeting results, because their contents and wording have been checked by the reviewers of the journal. Following an empirical procedure adapted from thesaurus building [5] we refer to classic thesaurus building methodology [1] where empirical and knowledge organization issues are discussed supported by decades of practical experience. We apply the principles of qualitative field research, especially the framework for inductive theory development [9]. According to the grounded theory framework, our ontology is a grounded model of its domain because all concepts are justified by and connected with their evidence in texts.

Because of the ontology’s high visibility for users and its central role for the future system’s practical value and acceptance, it is only consistent to expand the realm of user-centered system design [19, 22] from the user interface to the ontology, the system’s main knowledge representation. This follows from the core idea of user-centered system design that we have to think the system from its user interface, because for users, the system is what the interface presents. User roles with active participation are common in user-centered and participatory design. According to this practice, we try to integrate our users as responsible subject matter co-authors of the ontology.

The ontology is built up according to the ideal 13 steps procedure [5] presented in table 1, with some adaptations to practical conditions.

1	2–3 current relevant papers or book chapters are exploited to obtain an initial stock of concepts
2	concepts are supplemented with WordNet knowledge
3	if available, MeSH descriptors are added
4	the meaning of the concepts is made explicit and they are formalized and represented for the use of different players
5	users set up search scenarios
6	from user search scenarios queries are derived, the search engines are started
7	the found documents are summarized
8	the summarization results are integrated into the question/answer scenarios
9	summaries are checked for failures by physicians and technical team members
10	the knowledge representation is improved
11	agents are adapted or created
12	back to step 5 as often as needed
13	a new partial ontology is integrated into the existing one

Table 1. The step-by-step procedure of empirical ontology construction and evaluation [5]

The first adjustment was stimulated by a domain expert. Instead of exploiting small numbers of papers from each subdomain, he proposed choosing two sorts of papers for the whole domain: BMT papers from Blood, a core journal of the domain where current issues are discussed, and BMT educational papers from the American Society of Hematology

(ASH) which present fundamental topics in a tutorial style. We are now arriving at stage 5 of the procedure in table 1, with the restriction that our first round of scenario testing has not yet given rise to a WWW search.

Experience of thesaurus builders shows that concepts in literature may systematically differ from concepts asked for in user queries. Several reasons for this phenomenon are known, for instance different interests in information demand and information offer (with demand sometimes preceding offer) or in knowledge background, as with knowledge producers rooted in scientific theories and application-oriented users referring to commercial categories. With this in mind, we feed our ontology also from user query scenarios. With an additional knowledge base where domain experts can state their own knowledge, we arrive at three knowledge sources (see figure 2):

- Papers of a core journal and educational papers in BMT are the prevailing knowledge source.
- Text-based knowledge is supplemented by concepts from user scenarios.
- Concept records provided by domain experts are added.

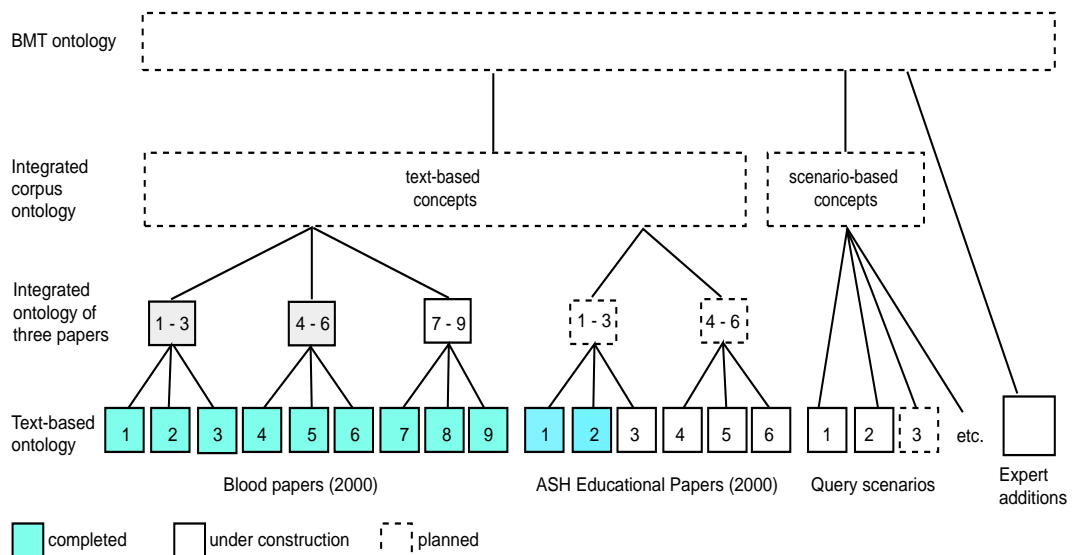


Fig. 2. The empirical structure of the BMT ontology

Figure 2 gives an overview of the empirical structure of the BMT ontology under construction. Partial ontologies with a light gray background are completed, planned ones have broken lines and clear backgrounds. All others with normal lines and white backgrounds are currently under construction.

Basic ontology records include the following main fields:

- concept
- synonyms
- hypernyms
- hyponyms
- description
- occurrences
- assertions

We add descriptions of concepts to the concept record, because when formulating questions, users must have the opportunity to check whether they have selected the appropriate concept. During knowledge acquisition from a paper all concept occurrences with their contexts, normally a sentence, are copied to the field ‘occurrences’. If these contexts convey valid knowledge, more concise statements are entered into the assertions field of the record. They are transformed into predicate logic expressions later on (see section 2.2).

The ontology is grounded in so far as all concepts are justified by and connected with their evidence in text. As soon as we integrate text-based ontologies, the bulky occurrence data are left behind, but they can be recovered from higher levels of ontology integration. For the sake of ease and safety, integration is done by bundles of three text-based ontologies at a time.

During integration, conceptualizations are checked and adapted if necessary, double records are removed, and statements found in the source databases are entered into the respective record in the integrated part of the ontology. If necessary and possible, the concept descriptions are improved. A return to the source paper is sometimes necessary for checking and changing conceptualizations.

When we glean concepts from user query scenarios, the empirical evidence is somewhat weaker than elsewhere. Often, the physicians’ scenario descriptions are very short, context information is poor or almost missing. To make user-defined scenarios fit for WWW retrieval, we have to modularize them and to fill them up with concepts from external sources. All concepts needed for stating usable scenarios are entered into the respective scenario ontology.

After illustrating the way the ontology is set up, we give a quantitative account of the concepts acquired so far. Table 2 shows the number of concepts derived from Blood (2000) papers. The scenario ontology of the first survey turn contains 211 concepts resulting from 28 scenarios. We have started lexical equivalences recently, the same is true for formalized contexts (see section 2.2).

Text-based Ontologies	Records	Integrated Ontologies	Records
Blood1	235	Blood1-3	479
Blood2	173		
Blood3	166		
Blood4	160		
Blood5	415	Blood4-6	781
Blood6	241		
Blood7	134		
Blood8	227		
Blood9	133		

Table 2. Overview of ontology records

Summit top, the upper model of the ontology (its first two levels are shown in figure 3), has been conceived deductively by drawing from the ontologies comprised in the metathesaurus of UMLS (Unified Medical Language System). Just adopting one of these classifications proved to be awkward. The MeSH classification for example does not fit our needs:

- The MeSH taxonomy is not strictly generic which is a prerequisite for our ontology with its strict isa-relations for inheritance management.
- The MeSH taxonomy is broad and its granularity does not suffice for our purposes.

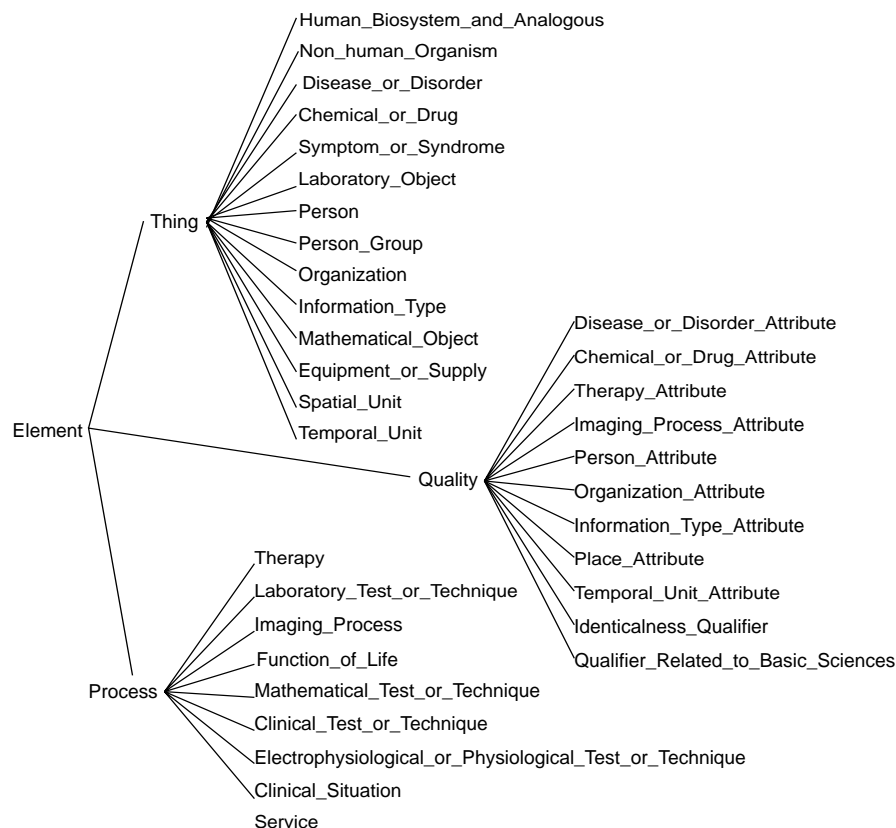


Fig. 3. The ontology top (SummIt top) in its current state

For the BMT taxonomy an expert selected those MeSH classes at the two upmost levels of this classification which fit the requests of our task. If requisite, they were reformulated so that the expression of generic relations was facilitated. We enhanced the classes by assigning 455 concepts to them and by inserting classes (e.g. clinical situation) if concepts could not be assigned to the existing taxonomy. If necessary, class designations and the internal structuring of classes were changed. The resulting classification was compared with other ontologies comprised in the metathesaurus of UMLS.

We also considered the Generalized Upper Model (GUM) [3] and the BigTop proposed by Sowa [20]. As we are using the ontology for text processing, we had an eye upon the classes that show up in natural language text and are recognized there by our parser³.

The preliminary top structure is useful because it sets an anchor to the generic hierarchies, but it is also often enough disputed by our empirical data. Experts of the domain verify concepts gathered by the text-based approach as well as the internal structure of the resulting classes. The taxonomy is adapted inductively as needed.

2.2 From Textual Context to First-Order Context Expressions

In medicine many statements are valid only if the limits of their scope are respected. Perhaps more than elsewhere we have to represent presuppositions. Furthermore, many of the semantic problems in NLP involve using a context to determine the meaning of an utterance.

³ The parser is provided by Conexor (<http://www.conexor.fi>).

To meet both the medical and the linguistic requirements, we apply the context formalization proposed by [11, 18]. Context expressions are composed of several propositions or predicates. While some of them set up the frame of thinking about the core propositions, the core predicates state the important points. A context expression as a whole must be true in domain expert eyes. Normally, it is the meaning equivalent of a sentence. When checking whether a context expression from input satisfies the conditions set by the query, summarizing agents first look for inconsistencies in context. If they find none, they examine the core of the context expression.

The context expressions and our predicate logic expressions are not physically stored in individual concept records, because almost all of them are shared by several concepts. Hence they are stored in and allocated from a central database. The information for recognizing concepts in running text (i.e. sets of textual equivalents) is stored in the central pool together with the predicates.

Context expressions are the main propositional knowledge representation format of the BMT ontology. They are obviously useful structures for stating medical knowledge because they limit the scope of an assertion. Context expressions assert that the proposition p is true ('ist') in the context c :

$$ist(c, p)$$

Both context and core propositions are predicate logic expressions. Following the example of Prolog, we accept only conjunctions and implications within context and core propositions. Disjunctions are expressed by alternative facts or rules. All predicates use ontology concepts as arguments.

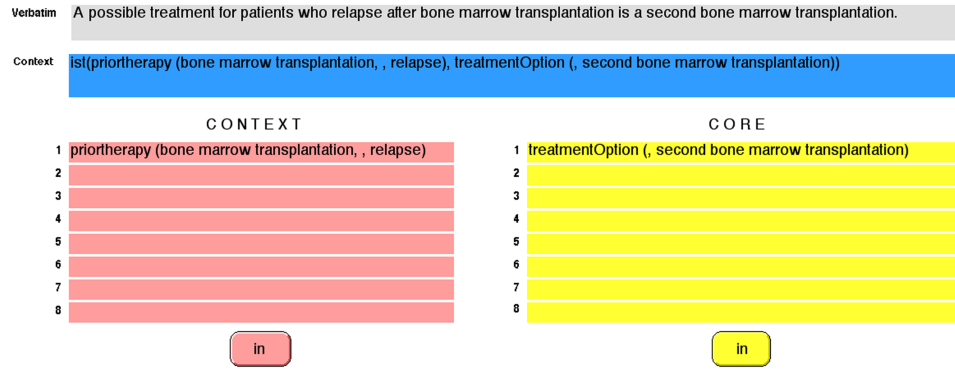


Fig. 4. A simple context expression is generated from context propositions on the left and core propositions on the right side

Figure 4 shows a context expression that is ready to enter the context database. It is accompanied by its formulation in natural language (top line). The context base accepts predicate expressions from the predicate base that are transferred either into the context area or into the core area. From the arriving context and core propositions, the context expression (second line) is generated.

Our context expressions are grounded upon knowledge found in the papers of our corpus. We extract contexts, often sentences, but also longer sequences if necessary for stating a context in stand-alone manner. While there are often quite explicit statements of medical knowledge, this is by no means always the case. We present a counterexample. It demonstrates that we can mine more and more relevant knowledge from a text by skilled interpretation. In Blood3 [19] we find:

DLI therapy would seem an acceptable alternative to second allogeneic BMT, however, as the morbidity and mortality of second BMT is high and prohibitive for many patients.

However, the incidence of acute or chronic GVHD and marrow aplasia may be acceptable compared with other potential treatment options such as second marrow transplantation.

However, given the excessive toxicity anticipated from second BMT within 1 year of the initial transplant, a trial of UDLI or other investigational therapy seems warranted.

Among many other things, we learn here that DLI therapy is an alternative to second allogeneic BMT. Doing a second bone marrow transplantation is presupposed three times. Readers can conclude this without bothering about the medical terms our example sentences are teeming with. The formulation in our ontology (cf. top line in figure 4) states the point as follows:

A possible treatment of patients who relapse after bone marrow transplantation is a second bone marrow transplantation.

Although this statement does not show up literally, it is a knowledge item whose validity is not confined to the paper at hand. It had to be derived by a competent human understander. She or he restates good assertions, elaborating implicit text knowledge if needed. Investing this intellectual work should enhance the quality of our ontology. Obviously, systems that can replace humans at this task are out of sight, only some technical support is feasible.

Textual formulations are stripped of rhetorical accessoires (example replacement rule: incidence of $X \Rightarrow X$) and prepared for later formalization. On their way to the context knowledge base they pass some normalization steps:

- Their concepts are standardized: only ontology concepts, no synonyms are accepted.
- They are reconstructed according to some syntax from the syntax base holding the syntaxes for predicate expressions. Thus they are normalized according to the number and kind of arguments.
- The resulting predicates are complemented with a natural language translation and stored in the predicate base for later reuse.
- The predicates are entered into the context expression, either into the core or the context position.

Stocks of context expressions are still modest at the time of writing: Some 243 of them are ready for use and approximately 340 predicates are equipped with equivalences.

2.3 Technical Implementation of the Ontology

From a technical point of view, ontology construction is supported by general-purpose tools. Source documents are PDF files and the selection of concepts from texts is facilitated by Conc⁴, a freeware concordancer. The ontology in the current state of development is stored in a set of FileMaker databases⁵. The Filemaker application environment supports ontology construction, also the development of the taxonomy and the formulation of context expressions with the underlying predicates and their argument structures (syntaxes).

⁴ The Conc concordancer is provided by the Summer Institute of Linguistics and is available at <http://www.sil.org/computing/conc/>.

⁵ FileMaker is a relational database server with a GUI for the Apple Macintosh environment. FileMaker is a registered trademark of FileMaker, Inc. ©1984-2000 (<http://www.filemaker.com>).

For consistency checking of the taxonomy, the corresponding FileMaker entries are imported into an XML database server (see below) and then exported to Graphviz, a free tool for displaying graph structures⁶. It helps to verify the correctness of generic relations.

Although there are existing tools for ontology acquisition like Ontolingua [6] and upcoming standards for ontology representation like OIL [7] we decided to use plain XML and developed our own DTD for ontology representation. Thus we can use the XML database server dbXML⁷ which provides a Java API for programming tasks and on that way retain the flexibility to import the ontology data into standardized tools later. Semantic checks may then be feasible or we will improve our tools in such a way that these checks are provided. When defining the DTD for the XML structure being used we restricted ourselves to the limitations propagated by the authors of SHOE [13] who demand semantic interoperability and therefore generality in the names of the XML tags. Thus, our tags are called `<concept>`, or `<proposition>` holding subtags like `<synonym>` or `<predicate_name>` instead of using tags like `<cancer>` and implementing concepts in a manner like `<cancer> leukemia </cancer>` as dismissed by the SHOE team.

3 Conclusion

We have presented an ontology for WWW summarization in Bone Marrow Transplantation. Due to the requirements of the task, the users, and the domain, the ontology comes with some extraordinary features. It is text-based — it qualifies as a grounded ontology — and it is user-centered. Currently, the ontology is still growing but its innovative features are set. We proceed now towards test and application. We have invested much intellectual work, work that is difficult for humans. As long as the core ontology is being constructed, we do not think that the human intellectual effort can be replaced by machine approaches without a quality loss that is not acceptable in a medical domain. Further extension of the ontology and its maintenance may be facilitated by semi-automatic ontology acquisition [16] and by concept input suggested by users. Compared with the development of thesauri, which may be much larger than the BMT ontology and as demanding in intellectual effort, our own intellectual investment is, after all, far from exiting.

Acknowledgements

Implementation of the SummIt-BMT system is supported by the German Science Foundation (DFG) under grant EN 186/6-1 and HE 2927/2-1, by the German Federal Ministry of Education and Research (bmbf) under grant 1701200, and by the Ministry of Science and Culture of Lower Saxony under grant 1999.384.

The authors would like to thank Andrea Köpcke and Juliane Topp for their contributions to this paper. Thanks also to the reviewers, especially to the one who confronted us with a long list of well-deliberated questions.

References

1. Aitchison, J., Gilchrist, A.: *Thesaurus Construction and Use: A Practical Manual*. 3rd edn. Aslib, London (1997)
2. Aussenac-Gilles, N., Biébow, B., Szulman, S.: *Corpus analysis for conceptual modelling*. EKAW 2000. http://www.irit.fr/ACTIVITES/EQ_SMI/GRACQ/WSEKAW2000/accepted.html (2000)

⁶ Graphviz is provided by AT&T (<http://www.research.att.com/sw/tools/graphviz/>).

⁷ dbXML has been made available by The dbXML Group (<http://www.dbxmlgroup.com>). The server is still under active development.

3. Bateman, J. A., Henschel, R., Rinaldi, F.: Generalized Upper Model.
<http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html> (1995)
4. Endres-Niggemeyer, B.: Summarizing Information. Springer, Berlin (1998)
5. Endres-Niggemeyer, B.: Empirical Methods for Ontology Engineering in Bone Marrow Transplantation. International Workshop on Ontological Engineering on the Global Information Infrastructure, Dagstuhl Castle, May 25, 1999.
<http://www.ik.fh-hannover.de/ik/person/ben/OntoBone.pdf> (1999)
6. Farquhar, A., Fikes, R., Pratt, W., Rice, J.; Collaborative Ontology Construction for Information Integration. Knowledge Systems Laboratory Department of Computer Science, KSL-95-63 (1995)
7. Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., Klein, M.: OIL in a Nutshell. In: Knowledge Acquisition, Modeling, and Management, In: Dieng, R. et al. (eds.): Proceedings of the European Knowledge Acquisition Conference (EKAW-2000). Lecture Notes in Artificial Intelligence, LNAI. Springer, Berlin (2000)
8. Fox, J., Das, S. K.: Safe and Sound: Artificial Intelligence in Hazardous Applications. MIT Press, Cambridge MA (2000)
9. Glaser, B. G., Strauss, A. L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. 11th edn. Aldine Atherton, New York (1980)
10. Golebiowska, J., Dieng-Kuntz, R., Corby, O., Mousseau, D.: Building and Exploiting Ontologies for an Automobile Project Memory, IJCAI 01 Workshop on Ontologies for Information Sharing.
<http://www.tzi.de/buster/IJCAIwp/Finals/golebiowska.pdf> (2001)
11. Guha, R. V.: Contexts: A Formalization and Some Applications. PhD thesis, Computer Science Department, Stanford University, Stanford, CA (1991). Also technical report ACT-CYC-423-91. Microelectronics and Computer Technology Corporation, Austin, TX (1991)
12. Hahn, U., Reimer, U.: Knowledge-Based Text Summarization: Saliency and generalization Operators for Knowledge Base Abstraction. In: Mani, I., Maybury M.T. (eds.): Advances in Automatic Text Summarization. MIT Press, Cambridge, Mass. (1999)
13. Heflin, J., Hendler, J., Luke, S.: SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), Dept. of Computer Science, University of Maryland at College Park (1999)
Also <http://www.cs.umd.edu/projects/plus/SHOE>
14. Hovy, E., Lin, C.-Y.: Automated Text Summarization in SUMMARIST. In: Mani, I., Maybury M.T. (eds.): Advances in Automatic Text Summarization. MIT Press, Cambridge, Mass. (1999)
15. Kindermann, C. Hoppe, T., Marwinski, D., Oertel, H., Paulus, O. K., Heimann, S., Schmiedel, A., Tolsdorf, R., Volle, P.: The MIHMA demonstrator application: bmt line. KIT-Report 132. Technische Universität Berlin, Berlin (1996)
Also <http://flp.cs.tu-berlin.de/~kit/reportliste/kitlistehtml.html>
16. Maedche, A., Staab, S.: Semi-automatic engineering of ontologies from text, Proceedings of the 12th International Conference on Software and Knowledge Engineering (SEKE 2000), Chicago, USA, 6-8 July 2000. http://www.aifb.uni-karlsruhe.de/WBS/publications/2000/seke_amaetal_2000.pdf (2000)
17. McCarthy, J., Buvač, S.: Formalizing Context (Expanded Notes). In: Aliseda, A., Glabbeek, R. van, Westerstohl, Dag (eds.): Computing Natural Language. Stanford University, Stanford (1997) 13–50. Also <http://www-formal.stanford.edu/buvac/formalizing-context.ps>
18. Norman, D. A., Draper, S. W. eds.: User Centered System Design. Erlbaum, London (1986)
19. Porter, D.L., Collins Jr, R. H., Hardy, C., Kernan, N. A., Drobyski, W. R., Giralt, S., Flowers, M. E. D., Casper, J., Leahey, A. , Parker, P. , Mick, R., Bate-Boyle, B., King, R., Antin, J. H.: Treatment of Relapsed Leukemia after Unrelated Donor Marrow Transplantation with Unrelated Donor Leukocyte Infusions. Blood Vol. 95(4) (2000) 1214–1221
20. Sowa, J. F.: Knowledge Representation: Logical, Philosophical, and Computational Foundations. Brooks Cole Publishing Co., Pacific Grove, CA (2000)
21. Winograd, T. (ed.): Bringing Design to Software. Addison-Wesley, Reading MA (1996)

Building Shared Ontologies for Terminology Integration

Gerhard Schuster and Heiner Stuckenschmidt

Center for Computing Technologies
University of Bremen, Germany

Abstract. We present an approach for developing shared ontologies which can be used to define terms from different vocabularies and to automatically translate them from one vocabulary into another. We first motivate the use of shared ontologies as a means for identifying semantic correspondences between terms and underline the need for a shared ontology. Then, the general steps of a specialized methodology for building such shared ontologies are described. We further illustrate the application of the methodology using a real-life example from the domain of geo-informatics.

1 Introduction

The exchange on information has become a crucial factor in today's economy. Many processes in business and administration involve different organizations that have to work together in order to reach a common goal. Further, the collection and maintenance of information is a time-consuming and costly effort that leads to the need of using existing information whenever possible. The most prominent example of an information repository accessible in principle is the World Wide Web with its millions of pages. However, the World Wide Web is also the best example of the problems that are connected with the exchange of information. Most of these problems are related to the heterogeneity of information sources. On the World Wide Web syntactic heterogeneity in terms of different encoding languages and formats is more or less solved by the standardization efforts of the World Wide Web Consortium W3C. However, the heterogeneity in the information itself is even more striking: using XML users are able to build their own data structures and terminologies. While the structures can be adopted almost without loss of information using schema information provided by document type definitions or XML schemas the use of different terminologies may lead to serious problems when the intuitive meaning of the elements from the terminology is not clear and the understanding of the information provider differs from that of the user.

We propose an approach for finding mappings between classes from different ontologies that combines the ideas of comparing class structures and definitions with the one of using common upper-level ontologies. Our approach relies on

the definition of a common terminology similar to a upper-level ontology. The elements of this common terminology are used to describe classes from different ontologies in terms of formal concept expressions. This formal model can be used to derive correspondences between classes on a well-founded logical level. The approach will be described in section 2. Our approach relies on the existence of an explicitly shared terminology that is expressive enough to form the basis of concept expressions for the classes of all ontologies to be compared. On the other hand, the terminology should be as small as possible in order to ease comparison and reduce the effort of building it. Therefore a specialized method is required to build such shared terminologies. Such a method will be presented in section 3 of this paper. It has to be more specific than existing proposals for ontology engineering methodologies (e.g. [11],[3],[4]), because it has to take the specific needs of the integration approach into account. We present this method as well as the general approach for relating classes using a real-life example that is still small enough to be comprehensive.

2 Ontology-Based Terminology Integration

Our approach to the semantic integration problem is based on the view that each information source serves as a context for the interpretation of the information contained therein. This view implies that an information entity can only be completely understood within its source unless we find ways to preserve the contextual information in the translation process. This claim has two implications:

1. We have to represent the context of an information entity given by its source
2. We have to use this contextual information to integrate an entity into the new context given by the target of the translation

We argued that contextual knowledge of an information entity can be represented by necessary and sufficient conditions for deciding whether an entity belongs to a certain class of objects [9]. Using these conditions the integration of an entity in a new context is equivalent with a classification that is based on its contextual knowledge. Details of this approach are given below.

2.1 Specifying Information Context

In information sources contextual knowledge is often hidden in type information. Most information sources are based on a data model describing classes, attributes and relations. Each entity within the information source is assigned to one of these categories we will refer to as 'concepts' in the sequel. Depending on the intended use of the information source each concept is assumed to serve a special function and to show special properties necessary for that function. Some of these properties will explicitly be contained in the information source other properties remain implicit because there is a silent agreement that

a property always holds. In order to support semantic translation we have to explicate these hidden assumptions by defining necessary and sufficient conditions an information entity has to fulfill in order to belong to that concept.

Necessary Conditions: Concepts are described by a set of necessary conditions in terms of values of some properties p_i . We write p_i^X to denote that the entity X shows property p_i . We claim that there are properties that are characteristic for a concept and can therefore always be observed for instances of that class. We write $\mathcal{N}_C = \{p_1, \dots, p_m\}$ to denote that the concept c has necessary conditions p_1, \dots, p_m . Assuming that class and property definitions always refer to the same entity X we get the following equation:

$$N^c \equiv c(X) \Rightarrow p_i^X \wedge \dots \wedge p_m^X \quad (1)$$

Sufficient Conditions: On the other hand, we assume that an entity automatically belongs to the concept c if it shows sufficient characteristic properties. We write $\mathcal{S}_C = \{p_1, \dots, p_n\}$ to denote that p_1, \dots, p_n are sufficient conditions indicating that X belongs to the concept c . We characterize the class c by the following equation:

$$S^c \equiv p_1^X \wedge \dots \wedge p_n^X \Rightarrow c(X) \quad (2)$$

The distinction between necessary and sufficient conditions for concept membership enables us to identify entities that definitely belong to a concept because they show all sufficient conditions. On the other hand, we can identify entities that clearly do not belong to the concept, because they do not fulfill the necessary conditions.

2.2 Context Transformation

Concepts identify common properties of their members by defining necessary conditions for a membership. A classification problem is characterized by the determination of membership relations between an object under consideration and a set of predefined concepts. The identification process starts with data about the object that has to be classified. This data is provided by so-called observation. In the course of the classification the observed data is matched against the necessary conditions provided by the class definitions leading to one or more classes. The match between observations and membership conditions is performed using knowledge that associates properties of objects with their class. This view on classification can be formalized in the following way [6]:

- Let C be a set of solution classes (in our case concept predicates $\{c_1, \dots, c_m\}$)
- Let O be a set of Observations (in our case the necessary conditions for concept membership $\{N^c | c \in C\}$)
- Let R be a set of classification rules (in our case sufficient conditions for class membership $\{S^c | c \in C\}$)

Then in principle a classification task is to find a solution class $c_i \in C$ in such a way, that

$$O \wedge R \Rightarrow c_i(X) \quad (3)$$

In terms of the definitions given above, semantic translation is equivalent to a re-classification of entities already classified in one semantic structure $C^S = \{c_1^S, \dots, c_n^S\}$ using another semantic structure $C^T = \{c_1^T, \dots, c_m^T\}$. The process of re-classification can be based upon the semantic characterizations given by both structures. The source structure provides the observations ($O = \{N^c | c \in C^S\}$), while solution classes and classification rules are provided by the target structure ($C = C^T, R = \{S^c | c \in C^T\}$). Using these definitions, a single information entity can be translated from one context into the other by finding a concept definition c_i^T in the target structure satisfying equation 3.

2.3 Support for the Integration Process

The considerations from the last section provide a theoretical foundation for semantic translation. However there are still many problems that have to be solved to put this approach to work. The most important question is how and what kind of context knowledge has to be considered in the translation process because the choice of the representation has major impacts on the classification method to choose and the expected results. Ontologies can play an important role in the translation process because their ability to explicate context knowledge can provide great support. In the following we analyze the roles different ontologies play in our translation approach and describe how they support the whole process of information integration.

The Role of Ontologies A closer look at the semantic translation approach described above reveals that different ontologies are used for different purposes within the approach. In order to get clear notions of these different roles we adopt the distinction made in [5]. Jasper and Uschold distinguish three roles an ontology can play in an application scenario, each associated with a level of application:

- L_0 : Operational data
- L_1 : Ontology
- L_2 : Ontology representation language

We will see that each of these roles occur within our framework. Each role is filled by a another kind of ontology with different extends of explication according to the specific requirements.

Operational Information that should be translated from one information source to another corresponds to L_0 . We argued that the real task is to determine the concept an information entity belongs to in a new context. So we rather translate type annotations than the information entity itself. This type information

already is an ontology in the sense of an explicit specification of a conceptualization, because we have to describe the concepts we want to translate. As a consequence we are already concerned with an ontology on the level of operational data. However this ontology does not show a large extend of explication, because it consists of a set of concept terms arranged in a simple taxonomy.

Specification of Contextual Knowledge is the basis for the translation of information entities. We use necessary and sufficient conditions for concept membership to specify contextual knowledge. This kind of context explication is a typical application of an ontology. The descriptions of necessary and sufficient conditions therefore is an ontology corresponding to level L_1 . It shows a larger extend of explication than the pure taxonomy of concept terms, because it explicates the intended meaning of these terms. Each information source to be integrated is supposed to be specified by such an ontology to enable us to use its contextual knowledge in the translation process.

Properties of Concept defining necessary and sufficient conditions serve as a common vocabulary used to build the ontologies of different information sources to be integrated. As such they can be seen as an ontology representation language corresponding to level L_2 . They have to be shared across all information sources to enable a classifier to check whether conditions are fulfilled. They explicate a common understanding of a basic vocabulary that is necessary to explain and exchange specialized vocabulary from different information sources. The extend of explication required from an ontology specifying properties largely depends on the complexity of the information to be translated and requirements on the efficiency of the translation. If complex information has to be translated once more complex property definitions may be used than in the case of simple information that has to be translated in real-time.

Process and Supporting Technologies In order to clarify the use of different ontologies we will now discuss the process of intelligent information integration that is implied by our approach. The process sketched below describes actors, supporting tools and knowledge items (i.e. ontologies) involved. Notice that although the approach described above translates only between two sources at a time, it is not limited to bilateral integration, because we do not use a hard-coded translator but a general classifier that will be able to integrate every information source owning a suitable semantic annotation.

Authoring of Shared Terminology Our approach relies on the use of a shared terminology in terms of properties used to define different concepts. This shared terminology has to be general enough to be used across all information sources to be integrated but specific enough to make meaningful definitions possible. Therefore the shared terminology will normally be built by an independent domain expert who is familiar with typical tasks and problems in a domain, but who is not concerned with a specific information source. As building a domain

ontology is a challenging task sufficient tool support has to be provided to build that ontology. A growing number of ontology editors exist [1]. The choice of a tool has to be based on the special needs of the domain to be modeled and the knowledge of the expert.

Annotation of Information Sources Once a common vocabulary exists, it can be used to annotate different information sources. In this case annotation means that the inherent concept hierarchy of an information source is extracted and each concept is described by necessary and sufficient conditions using the terminology built in step one. The result of this annotation process is an ontology of the information source to be integrated. The annotation will normally be done by the owner of an information source who wants to provide better access to his information. In order to enable the information owner to annotate his information he has to know about the right vocabulary to use. It will be beneficial to provide tool support also for this step. We need an annotation tool with different repositories of vocabularies according to different domains of interest.

Semantic Translation of Information Entities The only purpose of the steps described above was to lay a base for the actual translation step. The existence of ontologies for all information sources to be integrated enables the translator to work on these ontologies instead of treating real data. This way of using ontologies as surrogates for information sources has already been investigated in the context of information retrieval [12]. In that paper we showed that the search for interesting information can be enhanced by ontologies. Concerning semantic translation the use of ontologies as surrogates for information sources enables us to restrict the translation on the transformation of type information attached to an information entity by manipulating concept terms indicating the type of the entity. The new concept term describing the type of an information entity in the target information source is determined automatically by a classifier that uses ontologies of source and target structures as classification knowledge. This is possible, because both ontologies are based on the same basic vocabulary that has been built in the first step of the integration approach.

3 Building Shared Ontologies - An Overview

The integration process sketched above relies on the existence of a shared ontology suitable to define concepts from all terminologies to be integrated in sufficient detail. This requirement is a challenge with respect to ontology building. In order to support this difficult task, we propose a development methodology that is tailored to the purpose of building shared ontologies. In this section we give an overview of the development process. After describing a concrete integration problem (section 4) we will present a trial run through the methodology in section 5.

3.1 The Process

The proposed methodology is based on stepwise-refinement. It consists of five steps executed in sequence resulting in a partial specification of the shared ontology. The last step of each run is an evaluation step that triggers one of the previous steps in order to extend and refine the ontology if necessary.

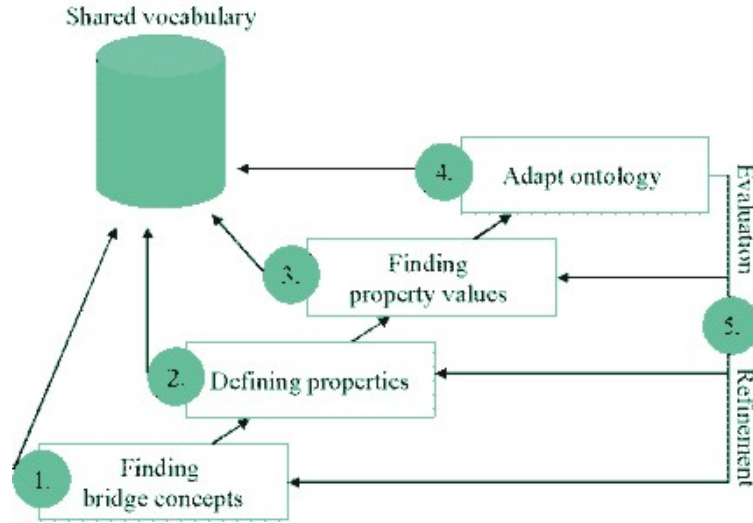


Fig. 1. Steps of the Development Process

Figure 1 illustrates the process model, the individual steps are briefly described below.

Step 1: Finding Bridge Concepts The first step is to examine the translation task. Asking the question "what do I want to translate?" leads to a concept that subsumes all classes from the source and destination systems. Because this concept makes a semantic translation from one source into another possible we call it bridge concept. While defining its properties and attribute values through the methodology we achieve the needed shared vocabulary. The most general bridge concept is "top", a concept that subsumes every other possible concept. For an exact classification it is recommended to choose the bridge concept as concrete as possible. If needed, more than one bridge concept can be defined to enable semantic translation.

Step 2: Definition of Properties The next step is defining properties that describe the chosen bridge concepts. A car, for instance, can be described through its color, its brand, its price, etc.

Step 3: Finding property values Once we have defined the properties, we search for values which can fill the attributes. These "fillers" are the main part of the shared vocabulary.

Step 4: Adapt ontology The strict usage of the methodology often shows some problems. During the first development cycles, the shared ontology will not be expressive enough to define all terms to be translated. So it may be necessary to adapt the ontology by building a special "support ontology" in order to revise the problem.

Step 5: Refine Definitions The introduced methodology follows the "evolving" life cycle. It allows the engineer to step back all the time to modify, add and remove ontology definitions, e.g. refining the bridge concept or integrate further taxonomies into the shared vocabulary.

Each of the steps modifies a different aspect of the shared ontology. While step 1 is concerned with the central concept definition, step 2 defines slots, step 3 integrates existing taxonomies, and step 4 generates application-specific taxonomies. This fact is useful in order to determine where to go back to if the evaluation step reveals the inability to describe a certain aspect of a terminology to be integrated.

3.2 Sources of Information

The use of the ontology to be built as a common basis for communication between systems makes it necessary to stay as closely as possible to a vocabulary and conceptualization of the domain that is widely accepted as a standard. In order to meet this requirement, we use several sources of information to build upon. These information sources are existing ontologies and thesauri as well as scientific classifications and data catalogues.

Upper-Level Ontologies are mainly used to find the bridge concept which acts as a template for the definition of all terms to be translated. In most cases, the bridge concept is obvious, however, the use of an upper level ontology provides us with a vocabulary which is partly standardized.

Scientific Classifications are another form of standards describing the conceptualization of a domain. Classifications like taxonomies of animals or plants are common knowledge which can be used to specify concepts from domain-specific ontologies.

Domain Thesauri contain typical terms used in an application domain, therefore they are a natural source for finding concept names for the shared ontology. Further, many thesauri contain at least free-text definitions of the terms included. These definitions provide guidance for the definition of

concepts.

Linguistic Thesauri are used to supplement information taken from domain-specific thesauri. In contrast to the specialized vocabulary defined in domain-specific thesauri, linguistic thesauri can be used to identify correspondences between terms found in different information sources. Especially, we use linguistic thesauri to expand the search for definitions of terms to their synonyms.

Data Catalogues finally contain the definitions of the terminology to be modeled. Therefore they define the concepts to be modeled and are the basis for evaluating the expressiveness of the shared ontology at a specific point in the modeling process.

In the course of the modeling process, we stick as closely as possible to the information from the sources mentioned above. Therefore the selection of these sources, though not discussed in this paper is already an important step when building a shared ontology.

4 An Example Problem

In order to illustrate our methodology for building shared ontologies we use a real-life example from the area of sharing geographic information. We define the integration task to be solved and describe the terminologies that are subject to integration. Both will be the basis for a detailed description of the ontology building process in the next section.

4.1 The Task to be Solved

The opening of geographical information systems (GIS) and the interoperability between these systems demands new requirements for the description of the underlying data [10]. GIS normally distinguish different types of spatial objects. Different standards exist specifying these object types. These standards are also called catalogues. Since there is more than one standard, these catalogues compete with each other. To date, no satisfactory solution has been found to integrate these catalogues.

In order to address the semantic translation problem we assume a scenario where the existing land administration database that is normally based on the ATKIS classification should be updated with new information extracted from satellite images of some area. Satellite images are normally analyzed using image processing techniques. These result in a segmentation of different areas which are classified according to the CORINE land-cover nomenclature, a standard for the segmentation and classification of satellite images. The process of updating the land administration system with this new data faces two main problems:

1. The boundaries of the objects in the database might differ from the boundaries determined on the satellite image.
2. The class information attached to areas on the satellite images and the type information in the land administration system do not match.

The first problem is clearly out of the scope of this report, but the second one is a perfect example of a terminology integration problem.

The use of ontologies for the representation of contextual knowledge gives us two options: (a) *integrated views* and (b) *verification*. An integrated view from the users perspective merges the data between the catalogues. This process can be seen as two layers which lay on top of each other. The second option gives user's the opportunity to verify ATKIS data with CORINE land cover data or vice versa.

A query interface – this could be an intelligent dialogue within a GIS system – sends its request to an inference engine. The inference engine builds up the actual knowledge base by using the ontologies of the concepts. The interesting part of the whole idea is that the inference engine can infer on the actual knowledge base and is therefore able to derive new knowledge which can be used for further questions.

4.2 The Information Sources

The ATKIS catalogue is an official information system in Germany. It is a project of the head surveying offices of all the German states. The working group offers digital landscape models with different scales from 1:25.000 up to 1:1.000.000 with a detailed documentation in corresponding object catalogues. We use the large scale catalogue ATKIS-OK-1000. This catalogue offers several types of objects including definitions of different types of areas. Figure 2 shows the different types of areas defined in the catalogue.

CORINE land cover is a part of the CORINE Programme the European Commission carried out from 1985 to 1990. The results are essentially of three types, corresponding to the three aims of the Programme: (a) an information system on the state of the environment in the European Community has been created (the CORINE system). It is composed of a series of data bases describing the environment in the European Community, as well as of data bases with background information. (b) Nomenclatures and methodologies were developed for carrying out the programs, which are now used as reference in the areas concerned at the Community level. (c) A systematic effort was made to concert activities with all the bodies involved in the production of environmental information especially at international level. The nomenclature developed in the CORINE Programme can be seen as another catalogue, because it also defines a taxonomy of area types (see figure 3) with a description of characteristic

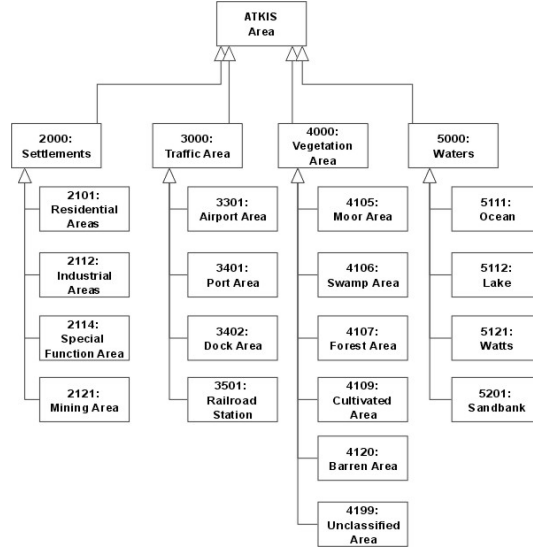


Fig. 2. Taxonomy of land-use types in the ATKIS-OK-1000 catalogue

properties of the different land types.

The taxonomies of land-use types in figures 2 and 3 illustrate the context problem mentioned in the introduction. The set of land types chosen for these catalogues are biased by their intended use: while the ATKIS catalogue is used to administrate human activities and their impact on land use in terms of buildings and other installations, the focus of the CORINE catalogues is on the state of the environment in terms of vegetation forms. Consequently, the ATKIS catalogue contains fine-grained distinctions between different types of areas used for human activities (i.e. different types of areas used for traffic and transportation) while natural areas are only distinguished very roughly. The CORINE taxonomy on the one hand contains many different kinds of natural areas (i.e. different types of cultivated areas) which are not further distinguished in the ATKIS catalogue. On the other hand, areas used for commerce and traffic are summarized in one type.

5 Building Terminologies - An Example

In this section we will show how a shared terminology and concept descriptions for the example problem can be developed using the methodology briefly described above.

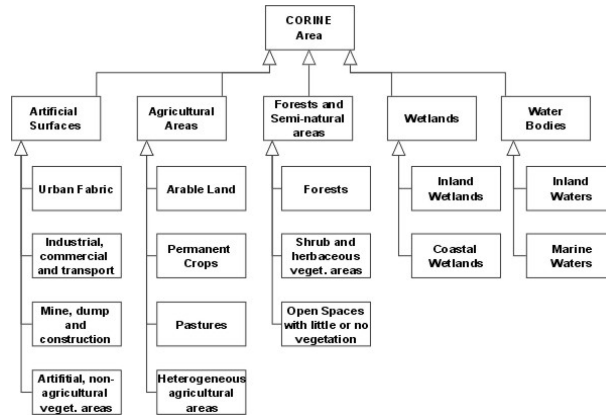


Fig. 3. A part of the taxonomy of land-use types in the CORINE land cover nomenclature

5.1 Information Sources

For this specific integration task we chose several sources of information to be used for guiding the development process. We briefly describe these sources in the following.

UpperCyc Ontology The UpperCyc, developed by Doug Lenat, CyCorp Inc. (<http://www.cyc.com>), is an upper-level ontology that captures approximately 3,000 terms of the most general concepts of human consensus reality. There is also a full Cyc knowledge base (KB) including a vast structure of more specific concepts descending below the UpperCyc, the so called top-level ontology. It contains millions of logical axioms – rules and other assertions – which specify constraints on the individual objects and classes found in the real world. Therefore the Upper Cyc ontology provides a sufficient common grounding for applications. It is possible to search the UpperCyc via internet: <http://www.cyc.com/cyc-2-1/find-constant.html>.

GEMET For the given scenario we choose the "General Multilingual Environmental Thesaurus (GEMET)", a polyhierarchically structured thesaurus which covers approximately 5.400 terms and their definitions organized by groups, themes, and terms. GEMET has been created by merging different national and international thesauri. Analysis and evaluation work of numerous international experts and organizations led to a core terminology of generalized environmental terms and definitions. GEMET ensures validated indexing and cataloguing of environmental information all over Europe. Where available, synonyms or alternate terms can be found likewise. The visualization tool for the GEMET is ThesShow, supporting the navigation through the GEMET database. It features, amongst others, extensive search, retrieval, and indexing functions.

Wordnet WordNet, developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller, is an on-line lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives, and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. You can query the WordNet through an online HTML form at <http://www.cogsci.princeton.edu/cgi-bin/webwn>.

Standard Taxonomies Scientific taxonomies can be found in many sources, like books or the internet. For this example we looked into the Google Web-directory (http://directory.google.com/Top/Science/Biology/Flora_and_Fauna) to obtain a classification of plant life. It is in no circumstances complete, but it satisfies our needs in this walkthrough.

5.2 Example Walkthrough

Based on the information described above we built up a first version of a shared ontology which should be used to solve the integration task mentioned in the last section. In this section we sketch the first development cycle of this ontology using the concrete modeling activities to illustrate the different steps of our methodology.

Step 1: Finding Bridge Concepts Looking at the given example scenario as described in section 4 it is quite obvious to choose a concept like "area" or "region", because all land-use classes are some kind of special "regions", or in other words, "region" subsums all land-use classes. We search for the term "region" in the "Upper-CYC" and get the following definition:

GeographicalRegion: A collection of spatial regions that include some piece of the surface of PlanetEarth. Each element of GeographicalRegion is a PartiallyTangible entity that may be represented on a map of the Earth. This includes both purely topographical regions like mountains and underwater spaces, and those defined by demographics, e.g., countries and cities [...]"

Figure 4 shows the hierarchical classification of the concept in the Upper-CYC. The definition fits very well, so finally we choose "Geographical Region" as our bridge concept. For further refinement we write it down in the OIL notation [2].

Class-def Geographical-Region

Step 2: Definition of Properties Now we have to find possible attributes for the bridge concept. We look for "Geographical Region" in the GEMET, but the search does not give any results. In that case the decomposition of the search phrase may give better results. For "Geography" and "Region" we get these definitions out of GEMET:

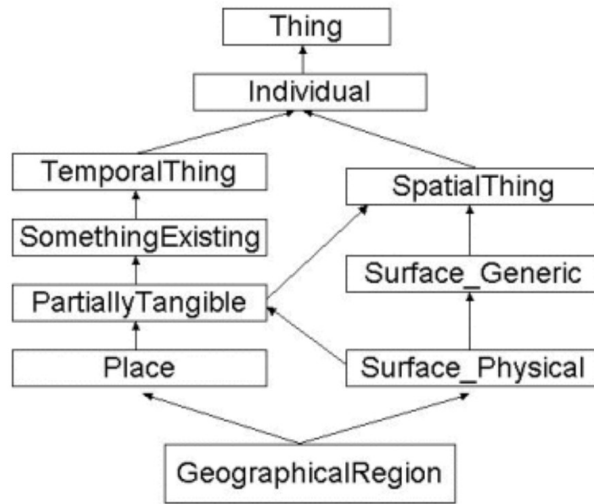


Fig. 4. Geographical Region in the Upper-CYC

Geography: "The study of the natural features of the earth's surface, comprising topography, climate, soil, vegetation, etc. and man's response to them."

Region: "A designated area or an administrative division of a city, county or larger geographical territory that is formulated according to some biological, political, economic or demographic criteria."

Here are some attributes clearly recognizable. For example "vegetation" is a biological criterion that defines a region, and it is also part of the scientific field geography. We update the bridge concept by defining a slot "vegetation" and adding it to the bridge concept.

```
Slot-def vegetation
  Domain Geographical-Region
```

```
Class-def Geographical-Region
```

Step 3: Integration of Standard Taxonomies To get possible "attribute values" or "filler" for the slot "vegetation", we take another look into GEMET. Vegetation is defined as:

"The plants of an area considered in general or as communities [...]; the total plant cover in a particular area or on the Earth as a whole."

We also check the synonym "flora", found in WordNet:

"The plant life characterizing a specific geographic region or environment."

The attribute "vegetation", respectively "flora", can be filled with terms out of plant life like "tree" or "rose" for instance. A good top concept is "plants", because many scientific taxonomies of plants exists. The Swedish botanist Carlous Linnaeus established 1753 a classification of plants. His work is considered the foundation of modern botanical nomenclature. In the Google Webdirectory we can access the plant kingdom with more than 10.000 entries online. We integrate this taxonomy into our vocabulary.

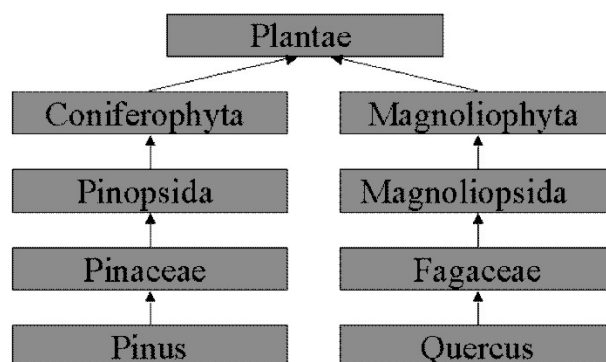


Fig. 5. Extract from scientific plant taxonomy

Now it is possible to describe classes from the land-use catalogues. The term "coniferous forest" in the CORINE context is defined as:

"Vegetation formation composed principally of trees, including shrub and bush understories, where coniferous species predominate."

In our vocabulary we find the term "Coniferophyta", comprising the conifers, which are trees or shrubs that bear their seeds in cones, without the protection of a fruit like angiosperms. This leads to the following OIL class:

```

class-def Coniferous_Forest
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Coniferophyta

```

The division Magnoliophyta of the plant kingdom consists of those organisms commonly called the flowering plants or angiosperms. The flowering plants are the source of all agricultural crops, cereal grains and grasses, garden and roadside weeds, familiar broad-leaved shrubs and trees, and most ornamentals. So, it is easy to describe the next CORINE class "broad leaved forest":

```
class-def Broad-leaved_Forest
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Magnoliophyta
```

A "mixed forest" in the CORINE nomenclature consists of conifers and broad-leaved trees.

```
class-def Mixed_Forest
  subclass-of Geographical-Region
  slot-constraint vegetation has-value Magnoliophyta
  slot-constraint vegetation has-value Coniferophyta
```

Step 4: Adapt vocabulary A closer look at the definition of the CORINE forest classes reveals that the classes are defined through the existence of trees and shrubs. Just using the term "Magnoliophyta" does not prevent the classification of a region covered with orchids as a broad-leaved forest (Orchidaceae is a subclass of Magnoliophyta). The mentioned taxonomy classifies plants by propagation, so there exists angiosperm and gymnosperm trees, shrubs and flowers. To handle this problem we need a more general distinction.

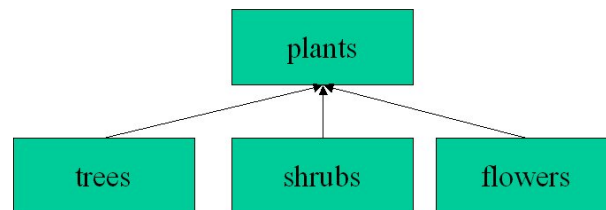


Fig. 6. Supplementary Plant Classification

Figure 6 shows a simple extension of the vocabulary that enables a more robust definition of the CORINE forest classes.

```
class-def Coniferous_Forest
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Coniferophyta and (trees or shrubs)
```

```
class-def Broad-leaved_Forest
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Magnoliophyta and (trees or shrubs)
```

```
class-def Mixed_Forest
  subclass-of Geographical-Region
  slot-constraint vegetation
```

```
has-value Coniferophyta and (trees or shrubs)
has-value Magnoliophyta and (trees or shrubs)
```

The shared vocabulary developed so far allows us to specify many different vegetation areas found in the land-use catalogues:

```
class-def Pastures
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Poaceae
```

```
class-def vineyards
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Vitis
```

```
class-def Rice_fields
  subclass-of Geographical-Region
  slot-constraint vegetation value-type Oryza
```

Step 5: Evaluation and Revision Not all CORINE landcover classes can be described after this first process cycle. "Mineral extraction sites", for instance, are defined as: "Areas with open-pit extraction of minerals (sandpits, quarries) or other minerals (opencast mines). Includes flooded gravel pits, except for river-bed extraction." No vegetation is mentioned, so the bridge concept must be refined. We go back to step 2 "defining properties" and search for another attribute. The definitions of "region" and "geography" show some anthropological aspects, like "man's response" or economic criteria. So we define a new slot "anthroposphere" and add it to our bridge concept:

```
slot-def anthroposphere
  Domain geographical-Region
```

```
slot-def vegetation
  Domain geographical-Region
```

```
class-def Geographical-Region
```

In GEMET exists the group "anthroposphere". One of its subclasses is "mining district", a district where mineral exploitation is performed. We integrate the partial taxonomy into the vocabulary (figure 4).

This special vocabulary can be used to simulate one-to-one mappings by using equality axioms. The CORINE class "mineral extraction sites" could be described as followed.

```
class-def Mineral-extraction-sites
  subclass-of Geographical-Region
  slot-constraint anthroposphere has-value mining-district
```

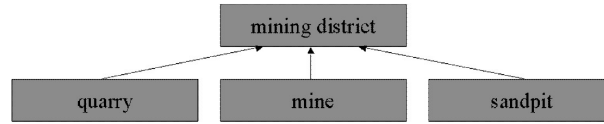



Fig. 7. Mining sites from the GEMET thesaurus

In a similar way we proceed iterating the process cycle until all terms from the two catalogue systems can be modeled as a specialization of the bridge concept. A further advantage of this strategy is the fact that the same process will be employed when additional terminologies are to be integrated as well. We cannot guarantee that the shared ontology also covers a new terminology, but we already provide guidance for the adaption of the ontology.

6 Discussion

We presented a method for building shared ontologies that can be used for terminology integration. The method was developed to support an approach to terminology integration by finding semantic correspondences we proposed in previous work [9, 7]. We first reviewed the integration approach and the role of shared ontologies in the approach. After giving an overview of the method, we introduced a real life problem and showed how the method can be used to successively develop a shared ontology for the problem. The example already shows that the method leads to better results than for example the hands-on approach described in [8]. However, there are still a number of open questions.

At the moment we only consider the case that the modifications to the ontology applied in each process cycle result in a strict refinement of the previous model. In our experiments this was always the case, however, in general it might be necessary to take back modeling decisions and revise parts of the ontology. In this case, we need adequate strategies for change management and versioning in order to avoid a loss of information. If parts of a previous model are removed, we have to track the impact on our possibility to define all terms that are subject to integration and we have to find ways to handle conflicts.

A second point concerns the evaluation of the results. At the moment we evaluate the ontology on the basis of its ability to be used to define concepts that represent terms from different terminologies to be integrated. As our goal is an automatic translation of terms, a decent evaluation should include translation trials. The use of a subsumption reasoner to prove semantic correspondences requires the model to be as complete as possible, a property that we do not check at the moment. Further, a measure is required that indicates, whether the model improves in the course of the process. Both topics, ontology revision and evaluation strategy will be addressed in future work.

References

1. A. Duineveld, R. Studer, M. Weiden, B. Kenepa, and R. Benjamis. A comparative study of ontological engineering tools, 1999.
2. D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. Oil in a nutshell. In *12th International Conference on Knowledge Engineering and Knowledge Management EKAW 2000*, Juan-les-Pins, France, 2000.
3. M. Fern'andez, A. G'omez-P'erez, and N. Juristo. Methontology: From ontological art towards ontological engineering, 1997.
4. M. Gruninger and M. Fox. Methodology for the design and evaluation of ontologies, 1995.
5. R. Jasper and M. Uschold. A framework for understanding and classifying ontology applications. In *12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*. University of Calgary/Stanford University, 1999.
6. M. Stefik. *Introduction to Knowledge Systems*. Morgan Kaufmann Publishers, San Francisco (CA), USA, 1995.
7. H. Stuckenschmidt. Using oil for intelligent information integration. In *Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods at ECAI 2000*, 2000.
8. H. Stuckenschmidt, F. van Harmelen, D. Fensel, M. Klein, and I. Horrocks. Catalogue integration: A case study in ontology-based semantic translation. Technical Report IR-474, Computer Science Department, Vrije Universiteit Amsterdam, 2000.
9. H. Stuckenschmidt and U. Visser. Semantic translation based on approximate reclassification. In *Workshop on Semantic Approximation, Granularity and Vagueness*, Breckenridge, Colorado, 2000.
10. H. Stuckenschmidt, U. Visser, G. Schuster, and Th. Voegelé. Ontologies for geographic information integration. In *Intelligent Methods in Environmental Protection: Special Aspects of Processing in Space and Time*. Center for Computing Technologies, University of Bremen, 1999.
11. M. Uschold. Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, 1996.
12. T. Voegelé, H. Stuckenschmidt, and U. Visser. Towards intelligent brokering of geo-information. In *Proceedings of the Urban Data Management Symposium*, Delft, 2000.