

# OntoMap: Portal for Upper-Level Ontologies

Atanas Kiryakov

OntoText Lab., Sirma AI Ltd.,

Hr. Botev 38A, 1000 Sofia, Bulgaria, naso@sirma.bg

and

Kiril Iv. Simov

Linguistic Modelling Laboratory, Bulgarian Academy of Sciences,

Acad. G. Bontchev Str. 25A, 1113 Sofia, Bulgaria, kivs@bgcict.acad.bg

and

Marin Dimitrov

OntoText Lab., Sirma AI Ltd.,

Hr. Botev 38A, 1000 Sofia, Bulgaria, marin@sirma.bg

**Abstract:** Currently the evaluation of feasibility of general-purpose ontologies and upper-level models is expensive mostly because of technical problems such as different representation formalisms and terminologies used. Additionally, there are no formal mappings between the upper-level ontologies that could ease any kind of studies and comparisons. We present the OntoMap Project (<http://www.OntoMap.org>), a project with the pragmatic goal to facilitate the access, understanding, and reuse of such resources. A semantic framework on conceptual level is implemented that is small and easy enough to be learned on-the-fly. We tried to design the framework so that it captures most of the semantics usually encoded in upper-level models. Technically, OntoMap is a web-site providing access to several upper-level ontologies and manual mapping between them.

**Categories & Descriptors:** I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods

**Keywords:** knowledge representation, mapping, ontology, semantic web

## 1. Introduction

Currently the evaluation of feasibility of general-purpose ontologies and upper-level models is pretty expensive mostly because of technical problems such as different representation formalisms and terminologies used. Additionally, there are no formal mappings between the upper-level ontologies that could ease any kind of studies and comparisons. As a result, the upper-level models are not widely used. In this paper we present a semantic framework on conceptual level

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*FOIS '01*, October 17-19, 2001, Ogunquit, Maine, USA.

Copyright 2001 ACM 1-58113-377-4/01/0010...\$5.00.

with the pragmatic goal to facilitate an easy access, understanding, and reuse of such resources. We tried to design the framework so that it captures most of the semantics usually encoded in upper-level models. The guideline was that the ability to easily evaluate, for example, 80% of their value will be good motivation for the developers of domain-specific ontologies, database schemas and other semantic models to use them as a basis. Technically OntoMap is a web-site (<http://www.OntoMap.org>) that provides access to upper-level ontologies and manual mappings between them. Currently it supports online browsing as well as export in DAML+OIL. The next step will be to provide the available resources in various other formats. In this way OntoMap will become part of the semantic web, i.e. machine-understandable rather than just human-readable.

The structure of the paper is as follows: first we present an introduction to the upper-level ontologies, knowledge representation languages and primitives. Section 3 describes the primitives used in the OntoMap project - the OntoMapO ontology. Section 4 focuses on the OntoMapO methodology for mapping concepts between ontologies. More technically section 5 discusses initial set of ontologies to be hosted and the mappings between them, the formats in which OntoMap will provide all the ontologies and mappings, and also the possible services. The next section demonstrates the OntoMap usability with a sample snapshot. Finally section 7 concludes the paper.

## **2. Upper-Level Ontologies**

Upper-level ontologies capture mostly concepts that are basic for the human understanding of the world. They are "grounded" in (supported by, wired to) the common sense that makes it difficult to formalize a strict definition for them. They represent prototypical knowledge using mainly taxonomic relations.

### **2.1 Domain-Specific vs. Upper-Level Ontologies**

This division seems to be a question of goal and scope of the developers of the ontology rather than a representational or management problem. However, there exists a significant real difference between the two types of ontologies. The domain-specific ontologies that are trying to capture, for example, a market segment or certain scientific area typically consist of well-defined concepts. For example, in the natural sciences (Mathematics, Physics, Chemistry, Biology, Medicine) the knowledge is easy to formalize because it is more or less systematic — it could be expressed using well-defined scientific terms. In such cases, the objects in the universe of discourse are either purely abstract or they are some idealized/simplified models of the real phenomena in the world.

### **2.2 Lexical Knowledge Bases**

The so-called lexical knowledge bases (LKB, such as WordNet) are lexicons, thesauri, or dictionaries that attempt to formalize the lexical semantics — the meanings of the words in one or more natural languages. Similar to the upper-level concepts, the meanings of the words are grounded in the common understanding of huge populations — there are no formal definitions, the words can bear a number of different meanings often based on associations, typical uses,

collocations, and prototypical knowledge. Going further, the meanings of many words are just primitive concepts. Some upper-level ontologies were developed on the basis of a LKB — such an example is the SENSUS ontology ([Knight and Luk 1994]). Other upper-level ontologies were developed in order to give formal semantics to a LKB — such an example is the EuroWordNet Top Ontology, [Vossen ed. 1999]. This is the reason to have a number of LKB semantic resources included in the set of ontologies to be hosted in OntoMap.

### 2.3 Philosophical diversity

The existence of several upper-level ontologies that disagree on the most basic concepts about the entities in the world demonstrates a significant philosophical diversity. The practical goals of the OntoMap project seems to require clarification of these basic discrepancies. Which properties of the entities in the world are the most basic ones? On which level of generality the differences disappear if they disappear at all?

Our understanding is that OntoMap should not try to choose the best upper-model or to produce a new one. The upper-level has to be chosen according to the specific application needs - the project goal is to facilitate this task.

### 2.4 Terminological Diversity

There is a number of different notions (or terminologies) that are currently used in the knowledge management community. The differences (both phraseological and conceptual) are rooted in the main paradigms in the knowledge representation. Here is a non-detailed overview of the most popular "languages" used by the ontologists:

- **concepts, relations, properties** — these are usually the terms inspired by the early **semantic networks**, mathematics and philosophy. *Concepts* are used to express any kind of static and cognitively autonomous semantic phenomena. They classify the entities in the domain of discourse (*instances* of the concept) — each entity either belongs to the interpretation of certain concept, or does not. The *properties* come to represent characteristics, aspects, or attributes of the entities, as well, as relations between them. They are further separated into *attributes* and *relations*.
- **classes, slots, facets and frames** — this is the frame-based terminology (**frames**). Here *classes* correspond to concepts while the notion for the instances remains the same. The *slots* correspond to the properties. Slots are further distinguished into *template-slots* and *own-slots*. The template-slots are defined on a class level — for example, Color is a template-slot for the class Car. In contrast, own-slots connect some values of the template-slots to certain instances of the class, say Colour(Ferri, Red). *Facets* are properties of the slots;
- **concepts, roles, individuals** — this is the terminology used in the so called description logics (DL), the descendants of the KL-One knowledge representation language. This paradigm is close to the one used in the semantic networks. It is developed to make them more precise on epistemological level. *roles* correspond to properties while *individuals* correspond to instances;

- classes, objects, attributes** - this is the terminology used in the **object-oriented** paradigm, mostly developed for the purposes of the software engineering. The *classes* correspond to the concepts while the *attributes* (data members) correspond to properties. Equivalent of the class-attributes are the *static data members*. The *objects* are always instances of certain class;
- collections, individuals, predicates, constants** - this is the terminology used by the *cyclists*, the people developing the Cyc knowledge base at Cyc Corp. Roughly, *collections* correspond to concepts, *individuals* to instances, and binary predicates (that are kind of collections themselves) correspond to properties. The *constants* are names of collections, individuals, or predicates.

In our view in order to provide a uniform representation of the ontologies and the mappings between them, a relatively simple meta-ontology (let us call it OntoMapO) of property types and relation-types should be defined. Before presenting OntoMapO we will give an overview of the related approaches.

## 2.5 The Conceptual Level of Unification

There are many attempts to resolve the terminological diversity by managing ontologies in a representation-independent fashion on the so called knowledge-level or conceptual level. Two of the most popular approaches are reported in [Gomez-Perez et al. 1998] (ODE) and [Maedche et al. 2000] (OntoEdit). Even kept within the frame-based terminology, the knowledge-model of Protégé-2000 (see [Noy et al. 2000]) is also a good example for a self-contained and well designed conceptualization that provides sufficient expressive power to capture ontologies encoded in different languages.

A comprehensive classification of the different kinds of properties is reported in [Guarino and Welty 2000] — according to different combinations of the meta-properties *indentivity*, *rigidity* and *dependence* it introduces seven different notions corresponding to "Concept" in ODE. The primitives used in Cyc (see [Cycorp 1997] and the previous sub-section) are interesting at least because the approach is proven in a really large-scale knowledge base.

## 3. OntoMapO: The OntoMap primitives

OntoMap is an attempt to use the minimal useful set of primitives. We are driven by the understanding that the oversimplification is not as fatal for the overall usability as a complex system of primitives could be. Thus we undertake an approach opposite to the one employed in Ontolingua, [Gruber 1991], following the rationale that even though many distinctions could be clearly defined in Ontolingua, many semantic-model developers have difficulties using them.

We developed a minimalistic meta-ontology that is as self-describing as possible. Thus, most of the primitives are defined just in terms of the rest of the OntoMapO primitives. Another primary consideration was to keep it decidable. Here is the point to mention that OntoMapO ontology could also be understood as a language. A simple language that provides some expressive power via single kind of expressions – binary relations between concepts. We are intentionally not providing specific syntax in order to keep it as representation independent

as possible. In the paper we use a LISP-like syntax to serialize the relations, however, it is obvious that many other notations could perform equally well.

### 3.1 Concepts, Relations, and Ontologies

*Concept* is the most basic primitive, so, we are leaving it to the reader's intuition. Just as a reference point the concepts could be compared to the constants in Cyc. The concepts could be related to each other by *binary relations*. Each binary relation has a type that is a concept. Each concept belongs to an ontology and, of course, there could be many different ontologies.

### 3.2 Instantiation in addition to inheritance

Our semantic framework was inspired Cyc ([Cycorp 1997]), Protégé-2000 ([Noy et al. 2000]), and RDFS ([W3C 1999]) representation models — in addition to the inheritance relations we also employ as a basic mechanism the instantiation. So, the concepts are not only described by their parents and children in the subsumption hierarchy but also form the classes that they belong to. The classes themselves are also concepts that could belong to other classes and so on. This way an infinite number of meta-levels could be defined.

We will use a simple set-theoretical semantics to explain the distinction between the inheritance and instantiation. Suppose that each concept is interpreted as a set of its instances. So, (**InstanceOf** I C) means that  $I \in C$ . In the same fashion (**ChildOf** C1 C2) means that  $C1 \subset C2$ . This interpretation has some pretty reasonable consequences:

- the inheritance relations are transitive — if (**ChildOf** C1 C2) and (**ChildOf** C2 C3) then (**ChildOf** C1 C3) holds. Really, from  $C1 \subset C2$  and  $C2 \subset C3$  it follows that  $C1 \subset C3$
- the instantiation is not-transitive — if  $I \in C$  and  $C \in \text{Meta}C$  it does **not** mean that  $I \in \text{Meta}C$
- the instantiation is transitive with respect to inheritance — if (**InstanceOf** I C1) and (**ChildOf** C1 C2) it follows that (**InstanceOf** I C2). Really, from  $I \in C1$  and  $C1 \subset C2$  it follows that  $I \in C2$

An obvious advantage of such extensive use of instantiation is that it makes the hierarchy less tangled by avoiding multiple-inheritance on many places. As a design principle, instantiation should be used to express non-sortal properties of the concepts (see [Guarino and Welty 2000]). For example, in OntoMapO (see below) we represent the transitivity of a relation type (say, **ChildOf**) via instantiation. The fact that certain relation is transitive does not determine its identity - it is just a rigid property, namely a Category for relations.

### 3.3 Relations

Each relation between two concepts is an instance of the concept representing its type. Let us extend the set-theoretical interpretation of our model — if (**RelA** B C) then the pair  $\langle B, C \rangle \in \text{RelA}$ . Suppose there are two concepts **RelA** and **RelB** that represent relation types and the first one inherits the second one (**ChildOf** **RelA** **RelB**). Our interpretation correctly predicts that **RelB** holds between all concepts where **RelA** holds. Let us show how it works:

- let us assume that there exist concepts **A** and **B** and there is a relation of type **RelA** between them (**RelA A B**)
- following our interpretation we can state that  $\langle A, B \rangle \in \text{RelA}$
- also (**ChildOf RelA RelB**) means that  $\text{RelA} \subset \text{RelB}$
- now it is obvious that  $\langle A, B \rangle \in \text{RelB}$ , which means that
- there is a relation of type **RelB** holding between the concepts **A** and **B**.

In **OntoMapO** all the concepts representing relation types should be instances of the **BinaryRel** concept or at least one of its children. Further, **OntoMap** inference engine considers a binary relation to be transitive iff it is an instance of **TransitiveRel** that is a child of **BinaryRel**. Examples for transitive relation are **ChildOf** and **Equivalent**. Analogously, a concept represents a symmetric relation type iff it is an instance of **SymmetricRel** — we can take **Inverse** relation (discussed below) as such example.

### 3.4 Each **OntoMapO** Relation Has an Inverse Relation

Another principle that we followed was to define an inverse relation for each of the **OntoMapO** relations except the symmetric ones, of course. The rationale behind this was twofold:

- to emphasize that the **OntoMap** relations (in contrast to the slot notion, for example) do not give any representational preference to the concept in the first place
- to make the relations easy to read and follow in both directions

So, **ChildOf** relation has its inverse **ParentOf** relation; **InstanceOf** is inverse to **ClassOf**. In order to keep some correspondence to the frame-based systems we defined **HasSlot** relation as an inverse to the **Domain** relation that could be defined between a relation type and the concept which instances could be first arguments of the relation. Analogously, **Reifies** is inverse to the **Range** relation that holds between a relation type and a concept which instances could be second arguments of the relation. Here are some real constraints that take place in **OntoMapO**:

- (**Domain Inverse BinaryRel**) and the equivalent statement that (**HasSlot BinaryRel Inverse**)
- (**Range Inverse BinaryRel**)
- (**Domain ChildOf Concept**) and its equivalent (**HasSlot Concept ChildOf**)

### 3.5 How Are the Predefined Relations Special

Let us call *sub-relations* of a relation **R** all its direct or indirect children as well as the relations that are equivalent to it or one of its sub-relations.

**OntoMap** considers as an equivalence relation each relation that is sub-relation **Equivalent**. In a similar fashion, all the sub-relations of **ChildOf** and **ParentOf** are treated as inheritance relations. Analogously, one relation is an instantiation relation iff it is a sub-relation of **InstanceOf** or **ParentOf** relations. Obviously, all the sub-relations of **Inverse** are properly interpreted as inversion by the **OntoMap** inference engine.

This approach makes the basic primitives that the OntoMap inference engine understands extensible. For example, when explaining Cyc's knowledge model to OntoMap it is easy to define that (`Equivalent #genls ChildOf`) — this way OntoMap automatically starts to understand this kind of Cyc inference relations without any need to translate them further.

### 3.6 The Hierarchy

The hierarchy below is basically an inheritance tree augmented with some instantiation information — after each concept name in brackets we have the classes it belongs to. Actually, just the informative classes are listed, i.e. the most specific ones. The inverse relations are indicated after each non-symmetric relation, to save space they are not separately presented.

```

Concept (Concept)
  BinaryRel (Concept)
    SymmetricRel (Concept)
    TransitiveRel (Concept)
Ontology (Concept)
  Context (Concept)
                                     meta-level above
-----
                                     language-level below
Inverse (SymmetricRel)
ChildOf (TransitiveRel) inverse: ParentOf
InstanceOf (BinaryRel) inverse: ClassOf
Domain (BinaryRel) inverse: HasSlot
Range (BinaryRel) inverse: Reifies
SimilarTo (SymmetricRel)
  Equivalent (TransitiveRel, SymmetricRel)
DisjointWith (SymmetricRel)
IsPartOf (TransitiveRel) inverse: HasPart
MadeOf (BinaryRel) inverse: SubstanceOf
MemberOf (BinaryRel) inverse: GroupOf

```

The above hierarchy does not include the ontology mapping primitives discussed in the next section. The separation into meta-level and language-level follows the scheme proposed in [Pan and Horrocks 2001]. However this separation corresponds more to the intuition of the above mentioned scheme. Formally, OntoMapO (as well as RDFS and Cyc) has unfixed meta-modelling architecture. This means that there can not be formally specified separate levels, `InstanceOf` relation can connect virtually any two concepts and the language itself does not constrain this. This is a major difference between OntoMapO and DAML+OIL.

## 4. Ontology-Mapping Primitives

There is no formal difference between the relations that can be used inside an ontology and those to be used for mapping of concepts in different ontologies. However there exist relations that are not intended to be used inside well defined ontology — those should be used for handling structural differences between different ontologies. For example, the (`TopInstance A B`) should be used when a concept `A` from one ontology exists as a concept `B` on the upper level of denotation in another ontology, i.e. (`ChildOf X A`) holds iff (`InstanceOf X B`) holds. Such design patterns should not be tolerated inside a single ontology. Here are the explanations for these relations:

- `MuchMoreSpecific` – the first concept is much more specific than the second one, transitive. Inverse to `MuchMoreGeneral` and a specification of `ChildOf`. Both (with the next relation) have to be used to "constrain" the meaning of a concept that has to be mapped in another ontology but has no equivalent or even similar concept there;
- `MuchMoreGeneral` – the first concept is much more general than the second one, transitive. Inverse to `MuchMoreSpecific` and a specification of `ParentOf`
- `TopInstance` – the first concept is the most general instance of the second one – a meta-concept. Inverse to `ExactClass` and a specification of `InstanceOf`
- `ExactClass` – the first concept is a meta-concept, the second concept is the most general instance of the first one. Inverse to `TopInstance` and a specification of `ClassOf`
- `ParentAsInstance` – the first concept is more general than all the instances of the second one that is a meta-concept. Inverse to `ChildAsClass`
- `ChildAsClass` – the first concept is a meta-concept (class), all its instances are more specific than the second concept. Inverse to `ParentAsInstance`

### 4.1 Representing Ontologies in OntoMap

When an ontology representation has a well defined conceptualization our approach is to map its primitives to the `OntoMapO` primitives. For example, importing Upper-Cyc Model ([Cycorp 1997]) we just defined that

```
(Equivalent # $genls ChildOf)
(Equivalent # $isa InstanceOf)
```

and so forth with the rest of the Cyc's relations. While `OntoMapO` interprets each relation that is equivalent or more specific than `ChildOf` as inheritance relation it starts perfectly understand the inheritance in Cyc.

Analogously importing Protégé-2000 ([Noy et al. 2000]) meta ontology we can establish that:

```
(Equivalent :DIRECT-SUPERCLASSES ChildOf)
(Equivalent :DIRECT-SUBCLASSES ParentOf)
(ChildOf :DIRECT-INSTANCES InstanceOf)
(ChildOf :DIRECT-TYPE ClassOf)
```

First, let us answer why `:DIRECT-TYPE` is more specific than `ClassOf` — in Protégé each concept could be instance just of a single class. This limitation



makes `:DIRECT-TYPE` more specific relation than `ClassOf`. Even with this complication, OntoMap will be able to interpret the instantiation as it is defined Protégé because `:DIRECT-TYPE` is a specification (sub-relation) of `ClassOf`, so `:DIRECT-TYPE` is an instantiation relation.

## 5. The Initial Set of Ontologies, Formats and Representations

The following ontologies will be hosted initially: Upper Cyc Ontology [UCYC]; EuroWordnet Top Ontology; EuroWordnet Clusters – the clusters of EWN base concepts classified by top concepts, an extension of ETOP; WordNet 1.5, 1.6 and 1.7 [WNUB5,6,7] unique beginners and top nouns; CORELEX; SIMPLE Core Ontology; MikroKosmos top-level [MKOSTOP]; SENSUS top-level [SENSTOP].

For each of the ontologies there will be available an "executive summary" as well as the most important documents about it (papers, reports, guides and so on), URLs. Of course, the original "distributives" provided by the creators will be also available. The following ontologies are already hosted on OntoMap: , EWNTOP, WNUB7, SENSTOP, and MKOSTOP.

Mappings between some of the ontologies will be provided in order to ensure an easier understanding and comparison between them. So, the ontologies hosted will form an inter-connected graph. Such mapping already exists between EWNTOP and UCYC (see [Kiryakov and Simov 2000]). The mapping between WNUB7 and MKOSTOP and the later two ontologies was recently developed. Some of the relations exist because of the nature of the ontologies:

- EWNCLUST and EWNTOP – the concepts of the former one are just conjunctions of those of the later one
- CLEX and WNUB5 – same as above
- WNUB5, WNUB6, and WNUB7 – there exist a mapping provided by the creators
- SENSTOP and UCYC – it is available as a part of the UCYC distributives as well as separately by the SENSUS developers.

The following mappings will be developed as a part of the project:

- EWNCLUST to UCYC
- CLEX to EWNCLUST (both directions)
- ETOP to SIMC
- WNUB7 to UCYC, EWNTOP, and SIMC

The mappings will be available in the same formats as the ontologies. Actually, each mapping could be seen as an extension of the target ontology. For example, the mapping between EWNTOP and UCYC can be considered as an extension of the UCYC with the concepts of EWNTOP that are connected appropriately to the UCYC constants (see [Kiryakov and Simov 2000]).

The ontologies in OntoMap project will be presented in a number of standard forms: **FIPA compliant Ontology Agent**, **PROLOG**, **KIF**, **DAML+OIL**, **RDFS**, **HTML** - an online ontology browser as well as static pages available for

download, **SQL** scripts for ORACLE and MS SQL Server, Ready-to-use files for **MS Access** (MDB), Online application server accessible via **CORBA**, **EJB**, **RMI**, and **SOAP**. At present, only the online ontology browser and the DAML+OIL export are implemented. After the end of the project, in parallel with the maintenance of the server we will be able to provide the following services for both domain specific and upper-level ontologies:

- Creating/loading an ontology in the OntoMap system and hosting it there. This way it will become accessible in all the formats/services supported. Also its conformance profile could be determined (see [Genesereth and Fikes 1998] and Unified Representation section). A security subsystem will be developed, so the proprietary ontologies will not be publicly available.
- Developing of mappings with ontologies that are already hosted in OntoMap. The mappings themselves will be also available in all the supported formats.

## 6. An Usability Example

Here is an example of how the OntoMap could help understanding a complex case in the Upper Cyc Ontology - the **#\$MeetingTakingPlace** constant. The comprehension comes from the following sources: it is deeply positioned in the tangled subsumption hierarchy; and also some important information is encoded via instantiation. The most readable representation in [Cycorp 1997] is:

The collection of human meeting events, in which #\$Persons gather intentionally at a location in order to communicate or share some experience; business is often transacted at such a meeting. Examples include: a particular conference, a business lunch, etc.

**isa:** #\$DefaultDisjointScriptType, #\$ScriptType, #\$TemporalObjectType  
**genls:** #\$SocialGathering  
some subsets: (16 unpublished subsets)

The underlined text represents hyper-references to descriptions of the appropriate constants. Below follows the standard view on the same concept provided by OntoMap:

**Concept:** **#\$MeetingTakingPlace** [UpperCyc]

**Gloss:** The collection of human meeting events, ...

**Super-concepts (parents):** #\$SocialGathering;

**Indirect:** #\$IntangibleIndividual, #\$CompositePhysicalAndMentalEvent, #\$TemporalThing, #\$PhysicalEvent, #\$MentalEvent, #\$Intangible, #\$Thing, #\$SpatialThing, #\$MentalActivity, #\$PurposefulAction, #\$Situation, #\$HumanActivity, #\$AnimalActivity, #\$Event, #\$Action, #\$Individual, #\$SocialOccurrence

**Indirect parents in other ontologies:**

Physical[EWN\_Top], Top[EWN\_Top], Mental[EWN\_Top],  
Social[EWN\_Top], 2ndOrderEntity[EWN\_Top]

**Instance of:** [#\\$DefaultDisjointScriptType](#), [#\\$TemporalObjectType](#)

**Indirect:** [#\\$Collection](#), [#\\$ObjectType](#), [#\\$Thing](#), [#\\$SituationType](#),  
[#\\$SetOrCollection](#), [#\\$Intangible](#),  
[#\\$MathematicalOrComputationalThing](#), [#\\$ScriptType](#)

**Sub-concepts (children):** none

**Direct instances:** none

**All direct relations:**

[#\\$genls](#): [#\\$SocialGathering](#)

[#\\$isa](#): [#\\$TemporalObjectType](#)

[#\\$isa](#): [#\\$DefaultDisjointScriptType](#)

This was a "snap-shot" of the current on-line interface of OntoMap. Pay attention to the fact that both indirect parents and classes are displayed and it is extremely useful — it requires serious efforts to reconstruct these indirect relations manually. Also, super-concepts in the EWN Top Ontology can be seen, and that provides a good impression about a possible position of [#\\$MeetingTakingPlace](#) there. So people that are familiar with EWN top can get an idea about the meaning of the Cyc constant.

## 7. Conclusion

The OntoMap project is still in an early phase that makes it difficult to evaluate it. The experience gained providing the Upper Cyc Ontology as MS Access database is encouraging — even though the original resource is available for a long time more than two hundred people have found it useful and have downloaded it in this format just for the last few months. We got a very positive feedback for another experiment of ours — developing a mapping between the Upper Cyc Ontology and the EuroWordNet Top Ontology and then providing it as a database as well as an online service. Even without significant theoretical innovation such facilitatory efforts seem to be important for the development of the semantic modeling community.

The first practical result of the project is a Java implementation of an inference engine that already supports the OntoMapO language — it is sound and complete with its support for inheritance, instantiation, inverse, transitive, and symmetric relations. The biggest ontology that we experimented with (Upper Cyc Ontology, about 3000 concepts) can be loaded in few seconds and than queried real-time.

## References

Cyc Ontology Guide: Introduction.

<http://www.cyc.com/cyc-2-1/intro-public.html>

Fensel, Dieter *Ontologies: Their Glory and the new bottlenecks they create*. Presentation on "Semantic Web Project Proposal" workshop, Vrije Universiteit Amsterdam (the Netherlands), Dec 8, 2000

<http://www.ontoweb.org/workshop/amsterdamdec8/>

- Genesereth, Michael R., and Fikes, Richard eds. *Knowledge Interchange Format draft proposed American National Standard (dpANS)*. NCITS.T2/98-004 <http://logic.stanford.edu/kif/>
- Gomez-Perez, A.; Fernandez, M.; Blazquez, M.; Garcia-Pinar, J. M. *Building Ontologies at the Knowledge Level using the Ontology Design Environment*. <http://delicias.dia.fi.upm.es/articulos/ode/ode.html>
- Gruber, Thomas R. *Ontolingua: A Mechanism to Support Portable Ontologies*. TR KSL 92-66, Knowledge System Laboratory, Stanford University, 1991.
- Guarino, Nicola; Welty, Christopher *A Formal Ontology of Properties*. In the Proc. of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France. R. Dieg and O. Corby (Eds.), LNAI 1937, pp. 97-112, Springer Verlag, 2000.
- Kiryakov, Atanas; Simov, Kiril Iv. *Mapping of EuroWordNet Top Ontology to Upper Cyc Ontology*. In: Proc. of "Ontologies and Text" workshop, during EKAW 2000. Juan-les-Pins, French Riviera, Oct. 2, 2000. <http://www.ontotext.com/publications/index.html>
- Knight, K. and Luk, S. *Building a Large Knowledge Base for Machine Translation*. Proc. of the American Association of Artificial Intelligence Conference AAAI-94. Seattle, WA, 1994.
- Maedche, A.; Schnurr, H.-P.; Staab, S.; and Studer, R. *Representation Language-Neutral Modeling of Ontologies*. In: Frank (ed.), Proc. of the German Workshop "Modellierung" 2000. Koblenz, Germany, April, 5-7, 2000.
- Noy, Natalya F.; Fergerson, Ray W.; Musen, Mark A. *The Knowledge Model of Protege-2000: Combining Interoperability and Flexibility*. In the Proc. of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France. R. Dieg and O. Corby (Eds.), LNAI 1937, pp. 97-112, Springer Verlag, 2000.
- Pan, Jeff; Horrocks, Ian *Metamodeling Architecture of Web Ontology Languages*. In the Proc. of International Semantic Web Working Symposium (SWWS), July 30 - August 1, 2001, Stanford University, California, USA
- Vossen, Piek (ed.) *EuroWordNet General Document Version 3, Final*, July 19, 1999. <http://www.hum.uva.nl/ewn/>
- World Wide Web Consortium; Brickley, Dan; Guha, R.V. (eds.) *Resource Description Framework (RDF) Schema Specification*, 1999. <http://www.w3.org/TR/1998/WD-rdf-schema/>
- Uschold, Mike; King, Martin; Moralee, Stuart; and Zorgios, Yannis *The Enterprise Ontology*, The Knowledge Engineering Review, 1998, Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate).
- Uschold, Mike *Converting an Informal Ontology into Ontolingua: Some Experiences*, Univ. Edinburgh, Artificial Intelligence Application Institute (AIAI), AIAI-TR-192, March 1996.