OntoMap – the Guide to the Upper-Level

Atanas K. Kiryakov, Marin Dimitrov
OntoText Lab., Sirma AI EOOD
Chr. Botev 38A, 1000 Sofia, Bulgaria, {naso,marin}@sirma.bg

Kiril Iv. Simov
Linguistic Modelling Laboratory, Bulgarian Academy of Sciences
Akad. G. Bontchev Str. 25A, 1113 Sofia, Bulgaria, kivs@bgcict.acad.bg

Abstract. The upper-level ontologies are theories that capture the most common concepts, which are relevant for many of the tasks involving knowledge extraction, representation, and reasoning. These ontologies use to represent the skeleton of the human common-sense in such a formal way that covers as much aspects (or ”dimensions”) of the knowledge as possible. Often the result is relatively complex and abstract philosophic theory.

Currently the evaluation of the feasibility of such ontologies is pretty expensive mostly because of technical problems such as different representations and terminologies used. Additionally, there are no formal mappings between the upper-level ontologies that could make easier their understanding, study, and comparison. As a result, the upper-level models are not widely used. We present OntoMap — a project with the pragmatic goal to facilitate an easy access, understanding, and reuse of such resources in order to make them useful for experts outside the ontology/knowledge engineering community.

A semantic framework on the so called conceptual level that is small and easy enough to be learned on-the-fly is designed and used for representation. Technically OntoMap is a web-site (http://www.ontomap.org) that provides access to upper-level ontologies and hand-crafted mappings between them. Currently it supports just on-line browsing and DAML+OIL export. The next step will be to provide the resources in various formats, including an application server giving an uniform access to the resources via OKBC. This way OntoMap will become part of the semantic web, i.e. machine-understandable rather than just human-readable.

1 Introduction

The structure of the paper is as follows: the next section makes an introduction to the Upper-Level Ontologies, including a principal comparison to the domain-specific ones, discussion on their relations with the lexical knowledge bases and some incompatibility issues. Section 3 discusses the representation languages and primitives with subsections about the most significant paradigms and a short overview of the approaches for their unification. Section 4 describes the primitives used in the OntoMap project – the OntoMapO ontology. Section 5 focuses on the OntoMapO methodology for mapping concepts between ontologies. More technically section 6 enumerates the formats in which OntoMap will provide all the ontologies and mappings. The initial set of ontologies to be hosted and the mappings between them are discussed in section 7. The next section 8 demonstrates the OntoMap usability with a sample snapshot. Section 9 provides an idea about the services that could be provided on top of the results of the project. The next section concludes the paper.

2 Upper-Level Ontologies

The 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 hard to formalize a strict definition for them. They represent the so called prototypical knowledge.

For example, what should be a formal KL-ONE-style or Frame-style definition of a ”table”. Most of the tables have 4 legs, however, there are pretty obvious exceptions for tables with three legs, single leg or even without anything to be considered as a leg. There could be also a ”serious” table with 6 legs. What should be the minimum and maximum cardinality for the slot/role leg? And what should be the type restriction? This is the reason to have most of the upper-level concepts being primitive in KL-ONE terms – they can only have partial definitions, some necessary conditions that involve other partially defined concepts. This is the practical reason to have the upper-level ontologies (for example, Upper Cyc Ontology, SENSUS, MikroKosmos) defined mainly in terms of taxonomic relations. An attempt to strongly use attributes in their definitions could be hard, expensive, and usually leads to involvement of default reasoning or other similar mechanisms that cause intractability.

2.1 Domain-Specific vs. Upper-Level Ontologies

This pseudo-dilemma seems to be mostly a question of goal and scope of the developers of the ontology rather than a representational or management problem. Of course, 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 usually 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 either they are some idealized/simplified models of the real phenomena in the world. For instance, a triangle is nothing more than a polygon with three angles.

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. Historically the LKBs and the upper-level ontologies seriously influenced each other. Some upper-level ontologies were developed on the basis of a LKB — such example is the SENSUS ontology ([10]). Other upper-level ontologies were developed in order to give formal semantics to a LKB — such an example is the EuroWordnet Top Ontology, [15]. This is the reason to have a number of LKB semantic resources included in the initial 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 OntoMap project is after seem to require clarification of these basic discrepancies. Which properties of the entities in the world are the most basic ones? What follows from different choices on this level? On which level of generality the differences disappear if they disappear at all? For example, the top of the MikroKosmos ontology (see [14]) demonstrates a typical top-level:

```
ALL
    PROPERTY
        ATTRIBUTE
        RELATION
    OBJECT
        SOCIAL-OBJECT
        PHYSICAL-OBJECT
        MENTAL-OBJECT
        INTANGIBLE-OBJECT
    EVENT
        SOCIAL-EVENT
        PHYSICAL-EVENT
        MENTAL-EVENT
```

However, it ignores the stuff/object (countable) distinction that is very basic in Cyc (see [2]) and other upper-level ontologies.

Our understanding is that OntoMap should not try to choose the best upper-model or to produce a new one. The upper-level have to be chosen according to the specific application – we just want to make the comparison easier.

2.4 Some Disadvantages of the Automatic Mapping

Automatic mapping or merging of ontologies is not involved in OntoMap — we start with the assumption that there either exist some hand-crafted mappings, or such can be developed. Even though it seems that automatic mapping could reduce the efforts, the typical heuristics employed (see [11], [1]) can have a very limited role in the case of upper-level ontologies because of a number of reasons:

- there is relatively small number of upper-level resources, because they are complex, expensive, and (potentially) more reusable than the domain-specific ones;

- they are more complex than the domain-specific ones, because they handle more abstract and partially defined concepts. Much of the semantics is represented just as a free text gloss, rather than as a formula in some knowledge representation formalism. So, in many cases the equivalence (or mapping) between two concepts could be judged only by interpretation of the glosses;

- the mappings between upper-level ontologies are more re-usable because the ontologies are more reusable;
- the quality of the mapping is extremely important because a mistake in the upper-levels of an ontology can have terrible effect on the lower levels.

3 Unified Representation Needed

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 make an overview of the related problems and approaches.

3.1 Terminological Diversity

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

- **concepts, relations, properties** — these are usually the terms inspired by the early semantic networks (let us use the abbreviation **SemNet** below), 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 — each entity either belongs to a certain concept's interpretation, either not. In other words, the information carried out by the concept is either true for the entity either not. The entities that belong to the interpretation of the concept are called *instances* of the concept. Typical concepts are Person, Food, Meeting, Idea. 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*. Typical representatives are: color, gender (attributes); loves, causes (relations).
- **classes, slots, facets and frames** — obviously, this is the frame-based terminology (to be referred as **Frames** below). Here *classes* correspond to the concepts while the notion for the instances remains the same. The *slots* (especially as they evolved in the last years) correspond to the properties. Slots are further distinguished into *template-slots* (class- or concept-attributes in SemNet) and *own-slots* (instance-attributes in SemNet). 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(Ferarri, Red). *Facets* are properties of the slots, for example, Domain, Range, Min-Cardinality, Documentation. Formally speaking, they are own-slots of the slots. There is no clear favorite for a single term corresponding to the facet notion in the rest of the paradigms;

- **concepts, roles, individuals** — this is the terminology used in the so called description logics (DL), the descendants of the KL-One knowledge representation language (CLASSIC, LOOM, KRIS, SHIQ). This paradigm is pretty close to the one used in the semantic networks. Strictly speaking, it is developed to make them more precise on the epistemological level. The *roles* correspond to the properties, the *individuals* correspond to the instances;

- **classes, objects, attributes** - this is the terminology used in the **object-oriented** paradigm (OO), mostly popular for the purposes of the software engineering. The *classes* correspond to the concepts while the *attributes* (also called 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, the *collections* correspond to concepts, *individuals* to the instances, and binary predicates (that are kind of collections themselves) correspond to properties. The *constants* are names of collections, individuals, or predicates.

3.2 The Conceptual Level

There are number of 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 [4] (ODE) and [12] (OntoEdit). Even sticking to the frame-based terminology the knowledge-model of Protégé-2000 (see [13]) 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 [7] — according to different combinations of the meta-properties *identity*, *rigidity* and *dependence* it introduces seven different notions corresponding to ”Concept” in ODE. The primitives used in Cyc (see [2] and the previous sub-section) are interesting at least because the approach is proven in a really large-scale knowledge base.

4 OntoMapO: The OntoMap Primitives

OntoMap is trying to use the minimal useful set of primitives. We were led by the understanding that the oversimplification is not that fatal for the overall usability as a complex system of primitives could be. We tried to design OntoMapO to capture most of the semantics usually encoded in upper-level models. The guideline was to give access to 80% of the ”knowledge” content of of the upper-level ontologies within 20% of the complexity of the representation needed to capture all of it.

We developed a minimalistic meta-ontology that is also as self-describing as possible. Thus, most of the primitives are defined just in terms of the rest of the OntoMapO primitives.

OntoMapO ontology could be also seen 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. Further in this paper we will use a LISP-like syntax to serialize the relations, however, it is obvious that many other notations (say XML) could perform equally well.

4.1 Comparison with Other Approaches

We will try first to give an impression about OntoMapO by quickly comparing it to two well known approaches for ontology representation.

4.1.1 OntoMap vs. Ontolingua

Each concept is represented in Ontolingua with some twenty slots, many of which are not obvious for people that do not understand frames. For example, if somebody wants to understand the definition of `Corporation` in the Enterprise Ontology as it is represented in Ontolingua (see [18] and [17]) s/he has to bother about the meaning of slots like `Set-Cardinality` and `Relation-Universe`.

We undertake an approach opposite to the one employed in Ontolingua, [5], following the rationale that even though many distinctions could be clearly defined in Ontolingua the most of the semantic-model developers cannot understand them. Our vision is that the database designers, for example, should not be expected to learn complex frame-based theories.

4.1.2 OntoMap vs. RDFS and DAML+OIL

OntoMapO is much similar to the RDFS. The equivalent for `rdfs:Class` is `Concept` in OntoMap and again there are two basic relations: instantiation and inheritance (see subsection Instantiation in Addition to Inheritance). An equivalent of `rdf:Property` in OntoMap is `BinaryRel`.

We will try to outline just the major differences:

- in OntoMap `rdfs:Resource` is missing, actually there is no difference between classes and resources – they are both considered as concepts (resp. `rdfs:Class` and `Concept`).
- as a consequence the property types (resp. `rdf:Property` or `BinaryRel`) are considered as a sub-class of the concepts.
- there is no special relation for `rdfs:subPropertyOf` – the sub-class relation (resp. `rdfs:subClassOf` and `ChildOf`) is used for this purpose.

Similarly to the RDF triples, the basic expressive primitive in OntoMap are directed binary relations between the concepts that are labeled with other concepts.

The DAML+OIL language (see [8]) can be seen as extension of RDFS. OntoMapO is much similar and basically simpler than DAML+OIL. It is missing class expressions (no enumeration, boolean expressions, and property restrictions), Unique and Ambiguous properties. However, it is not the case that OntoMapO is a sub-language of DAML+OIL, in addition it has number of primitives for mapping (TopInstance, ExactClass, ParentAsInstance, and ChildAsClass) and meronymy (PartOf, MemberOf, and SubstanceOf).

4.2 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.

4.3 Instantiation in Addition to Inheritance

Our semantic framework got some inspiration from Cyc, Protégé-2000, and RDFS representation models (see [2], [13], and [16]) — 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 from 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 $\mathtt{I} \in \mathtt{C}$. In the same fashion `(ChildOf C1 C2)` means that $\mathtt{C1} \subset \mathtt{C2}$. This interpretation has some pretty reasonable consequences:

- the inheritance relations are transitive — if `(ChildOf C1 C2)` and `(ChildOf C2 C3)` it follows that `(ChildOf C1 C3)`. Really, from $\mathtt{C1} \subset \mathtt{C2}$ and $\mathtt{C2} \subset \mathtt{C3}$ it follows that $\mathtt{C1} \subset \mathtt{C3}$

- the instantiation is not-transitive — if $\mathtt{I} \in \mathtt{C}$ and $\mathtt{C} \in \mathtt{MetaC}$ it does NOT follows that $\mathtt{I} \in \mathtt{MetaC}$

- the instantiation is transitive with respect to inheritance — if `(InstanceOf I C1)` and `(ChildOf C1 C2)` it follows that `(InstanceOf I C2)`. Really, from $\mathtt{I} \in \mathtt{C1}$ and $\mathtt{C1} \subset \mathtt{C2}$ it follows that $\mathtt{I} \in \mathtt{C2}$

An obvious advantage of such extensive use of instantiation is that it makes the hierarchy less tangled avoiding multiple-inheritance on many places. As a design principle, instantiation should be used to express non-sortal properties of the concepts (see [7]). For example, in OntoMapO (see below) we represent the transitivity of a relation type (say, `ChildOf`) via instantiation. It is because the fact that certain relation is transitive does not determines its identity — it is just a rigid property, namely a Category for relations.

4.4 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)` than the pair $< \mathtt{B}, \mathtt{C} > \in \mathtt{RelA}$. 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 have concepts `A` and `B` and there is a relation of type `RelA` between them `(RelA A B)`

- following our interpretation we can state that $< \mathtt{A}, \mathtt{B} > \in \mathtt{RelA}$
- also `(ChildOf RelA RelB)` means that $\mathtt{RelA} \subset \mathtt{RelB}$
- now it is obvious that $< \mathtt{A}, \mathtt{B} > \in \mathtt{RelB}$, that means that
- there is a relation of type `RelB` 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.

4.5 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 two-fold:

- to emphasize that the OntoMap relations (in contrast to the slot notion, for example) does not give any representational preference to the concept in the first place
- two 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)`, equivalently `(HasSlot Concept ChildOf)`

4.6 How Are the Predefined Relations Special

Let us call *variants* of a relation `R` all its direct or indirect children (i.e. sub-relations or sub-properties) as well as the relations that are equivalent to it or one of its children.

OntoMap considers as an equivalence relation each relation that is variant of `Equivalent`. In a similar fashion, all the variants 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 `ClassOf` relations. Obviously, all the sub-relations of `Inverse` are properly interpreted as inversion by the OntoMap inference engine.

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

There is an interesting implementation issue related to this extensibility. In order to infer all the inheritance relations, the engine should know all the equivalence relations (to be able to detect the variants of `ChildOf` and `ParentOf`). However, the opposite is also true. The appropriate algorithms were implemented.

4.7 The Hierarchy

The hierarchy below is basically an inheritance tree augmented with some instantiation information – after each concept name in brackets we have the (most specific) classes it belongs to.

```
Top (Concept)
   Concept (Concept)
      BinaryRel (Concept)
         TransitiveRel (Concept)
         SymmetricRel (Concept)
   Ontology (Concept)
      Context (Concept)
   ChildOf (TransitiveRel)
      MuchMoreSpecific (TransitiveRel)
   ParentOf (TransitiveRel)
      MuchMoreGeneral (TransitiveRel)
   ClassOf (BinaryRel)
      ExactClassOf (BynaryRel)
   InstanceOf (BinaryRel)
      TopInstanceOf (BinaryRel)
   SimilarTo (TransitiveRel, SymmetricRel)
      Equivalent(TransitiveRel,SymmetricRel)
   Inverse (SymmetricRel)
   ChildAsClass (BinaryRel)
   ParentAsInstance (BynaryRel)
   Domain (BinaryRel)
   Range (BinaryRel)
   HasSlot (BinaryRel)
   Reifies (BinaryRel)
   DisjointWith (BinaryRel)
   IsPartOf (TransitiveRel)
   HasPart (TransitiveRel)
   MadeOf (BinaryRel)
   SubstanceOf (BinaryRel)
   MemberOf (BinaryRel)
   GroupOf (BinaryRel)
```

The full definition of OntoMapO in DAML+OIL (version from March, 2001) together with descriptions of each of the concepts is available at
`http://www.ontomap.org/2001/07/ontomapo`

5 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 are number of relations that are not expected 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 follow explanations for these relations:

- `MuchMoreSpecific`, `MuchMoreGeneral` – the first concept is much more specific (resp. general) than the second one. Both are transitive and inverse to each other and have to be used to ”constrain” the meaning of a concept that has not equivalent or even similar concept in another ontology;
- `TopInstance` – the first concept is the most general instance of the second one. Inverse to `ExactClass`;
- `ExactClass` – the first concept is a kind of meta-concept, the second concept is the most general instance of the first one. Inverse to `TopInstance`;
- `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`

5.1 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 ([2]) we just defined that

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

an 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 ([13]) 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.

6 Formats and Representations

Publicly available ontologies (or parts of them) will be presented in a number of standard forms:

- PROLOG and KIF;
- DAML-OIL (already available);
- 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 (an RPC protocol based on XML.)

  At present, the only the online ontology browser is implemented.

7 The Initial Set of Ontologies

The following ontologies will be hosted initially:

- Upper Cyc Ontology [UCYC]
- EuroWordnet Top Ontology [EWNTOP]
- EuroWordnet Clusters [EWNCLUST] - the clusters of EWN base concepts classified by top concepts. An extension of ETOP
- WordNet 1.5 unique beginners [WNUB5]
- WordNet 1.6 unique beginners [WNUB6]
- WordNet 1.7 unique beginners [WNUB7]
- CORELEX [CLEX] - made on top of WNUB5
- 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: UCYC, EWNTOP, WNUB7, and MKOSTOP. Clear candidates for hosting are (or will be) also OpenCyc, any results of the SUO effort, and the Simple Core Ontology.

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 interconnected graph. Such mapping already exists between EWNTOP and UCYC (see [9]). The mapping between WNUB7 and MKOSTOP and the later two ontologies was recently developed. Some of the relations in the graph exist because of the nature of the ontologies:

- EWNCLUST and ETOP - the concepts of the former one are just conjunctions of those of the later one
- CLEX and WNUB5 - same as above
- WNUB5 and WNUB6 - 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)
- WNUB7 to UCYC
- WNUB7 to EWNTOP

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 [9]). No ontologies and upper-level models will be developed under the project instead of the OntoMapO meta-ontology mentioned above.

8 One Usability Example

Here follows 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 two sources: it is deeply positioned in the tangled subsumption hierarchy; and also some important information is encoded via instantiation. The most readable representation in [2] 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, 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.

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 that both indirect parents and classes are displayed that is extremely useful — it requires serious efforts to reconstruct this indirect relations manually. Also, super-concepts in the EuroWordnet (EWN) Top Ontology can be seen, that provides a good impression about a possible position of `#$MeetingTakingPlace` there. These way people that are familiar with EWN top can get an idea about the meaning of the Cyc constant.

9 Ontology Unification Services

In parallel with the maintenance of the server and updates to the content we will be able to provide the following services for both domain specific and upper-level ontologies:

- Loading the ontology in OntoMap and hosting it there. This way it will become accessible in all the formats supported. Also its conformance profile could be determined (see [3] and Unified Representation section). A security subsystem will be developed, so the proprietary ontologies will not be publicly available.
- Developing of mappings to ontologies that are already hosted in OntoMap. The mappings themselves will be also available in all the supported formats.

10 Conclusion

OntoMap project is still in an early phase that makes it hard to evaluate it. The experience gathered 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 found it useful and downloaded it in this shape just for one year. 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 interesting result of the project is a Java-written 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 in real-time. The inference engine is used for support of an on-line ontology browser that is publicly available at `http:\\www.ontomap.org`. Apart from the engineering work needed to make available the remaining functionality of OntoMap portal we are constantly working on evaluation and development of the semantic framework – OntoMapO.

References

[1] Campbell, A.E. and Shapiro, S.C., *Algorithms for Ontological Mediation*, Technical Report 98-03, Dep. of CS and Engineering, State Univ. of New York at Buffalo, 1998.

[2] Cyc Ontology Guide: Introduction.
`http://www.cyc.com/cyc-2-1/intro-public.html`

[3] 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/

[4] 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

[5] Gruber, Thomas R. *Ontolingua: A Mechanism to Support Portable Ontologies.* Technical Report KSL 92-66, Knowledge System Laboratory, Stanford University, 1991.

[6] Gruber, Thomas R. *Toward Principles for the Design of Ontologies Used for Knowledge Sharing.* Technical Report KSL 93-04, Knowledge System Laboratory, Stanford University, 1993.

[7] Guarino, Nicola; Welty, Christopher *A Formal Ontology of Properties.* In the Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France. R. Dieg and O. Corby (Eds.): EKAW 2000, LNAI 1937, pp. 97-112, Springer Verlag, 2000.

[8] Ian Horrocks, Frank van Harmelen, Peter Patel-Schneider (eds.) *DAML+OIL (March 2001)* `http://www.daml.org/2001/03/daml+oil-index.html`

[9] Kiryakov, Atanas; Simov, Kiril Iv. *Mapping of EuroWordnet Top Ontology to Upper Cyc Ontology.* In: Proceedings of "Ontologies and Text" workshop, during EKAW 2000. Juan-les-Pins, French Riviera, Oct. 2, 2000. `http://www.ontotext.com/publications/ index.html# KiryakovSimov2000b`

[10] Knight, K. and Luk, S. *Building a Large Knowledge Base for Machine Translation.* Proceedings of the American Association of Artificial Intelligence Conference AAAI-94. Seattle, WA, 1994.

[11] McGuinness, Deborah L., Richard Fikes, James Rice, and Steve Wilder, *An Environment for Merging and Testing Large Ontologies*, Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000). Breckenridge, Colorado, USA. April 12-15, 2000.

[12] Maedche, A.; Schnurr, H.-P.; Staab, S.; and Studer, R. *Representation Language-Neutral Modeling of Ontologies.* In: Frank (ed.), Proceedings of the German Workshop "Modellierung" 2000. Koblenz, Germany, April, 5-7, 2000.

[13] Noy, Natalya F.; Fergerson, Ray W.; Musen, Mark A. *The Knowledge Model of Protege-2000: Combining Interoperability and Flexibility.* In the Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, France. R. Dieg and O. Corby (Eds.): EKAW 2000, LNAI 1937, pp. 97-112, Springer Verlag, 2000.

[14] Ortiz, Antonio Moreno. *Managing conceptual and terminological information in a user-friendly environment.* In: Proceedings of "OntoLex 2000: Ontologies and Lexical Knowledge Bases", Sozopol, Sept. 8-10, 2000. (to appear)

[15] Vossen, Piek (ed.) *EuroWordNet General Document Version 3, Final*, July 19, 1999. `http://www.hum.uva.nl/ ewn/`

[16] 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/`

[17] 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).

[18] Uschold, Mike *Converting an Informal Ontology into Ontolingua: Some Experiences*, Univ. Edinburgh, Artificial Intelligence Application Institute (AIAI), AIAI-TR-192, March 1996.