The UMLS® Semantic Network and the Semantic Web

Vipul Kashyap, Ph.D.
National Library of Medicine, Bethesda, Maryland
kashyap@nlm.nih.gov

The Unified Medical Language System® (UMLS®) , an extensive source of biomedical knowledge developed and maintained by the US National Library of Medicine (NLM) is being currently used in a wide variety of biomedical applications. The Semantic Network, a component of the UMLS is a structured description of core biomedical knowledge consisting of well defined semantic types and relationships between them. We investigate the expressiveness of DAML+OIL, a markup language proposed for ontologies on the Semantic Web, for representing the knowledge contained in the Semantic Network. Requirements specific to the Semantic Network, such as polymorphic relationships and blocking relationship inheritance are discussed and approaches to represent these in DAML+OIL are presented. Finally, conclusions are presented along with a discussion of ongoing and future work.

INTRODUCTION

The Unified Medical Language System® (UMLS®) project was initiated in 1986 by the U.S. National Library of Medicine (NLM). Its goal is to help health professionals and researchers use biomedical information from different sources[1]. It consists of three main knowledge repositories: (a) ***The UMLS Metathesaurus***, which provides a common structure for more than 95 source biomedical vocabularies. It is organized by concept, which is a cluster of terms (e.g., synonyms, lexical variants, translations) with the same meaning. (b) ***The UMLS Semantic Network***[2], which categorizes these concepts through semantic types and relationships. (c) ***The SPECIALIST lexicon*** contains over 30,000 English words, including many biomedical terms. Information for each entry, including base form, spelling variants, syntactic category, inflectional variation of nouns and conjugation of verbs, is used by the lexical tools[11]. The 2002 version of the Metathesaurus contains 871,584 concepts named by 2.1 million terms. It also includes inter-concept relationships across multiple vocabularies, concept categorization, and information on concept co-occurrence in MEDLINE.

The UMLS Semantic Network is highly suited for representation using DAML+OIL[5] constructs as it has a rich semantic structure and an underlying meta-model consistent with the DAML+OIL specification. In this paper, we investigate the expressiveness of DAML+OIL constructs for representing the knowledge contained in the Semantic Network. The results of this work will also be applied to the UMLS Metathesaurus.

DAML+OIL: AN ONTOLOGY LANGUAGE FOR THE SEMANTIC WEB

The recognition of the key role that ontologies are likely to play in the future of the Web has led to the extension of Web markup languages in order to facilitate content description and the development of web ontologies, e.g., XML Schema[7], RDF[4] and RDF Schema[8]. However, more expressive power is both necessary and desirable in order to describe data in sufficient detail, and enable automated reasoning, e.g., determine semantic relationships between syntactically different terms. The DAML+OIL language[5] is designed to describe the *structure* of a domain. It takes an object oriented approach, with the structure of the domain being described in terms of *classes* and *properties*. An ontology consists of a set of *axioms* that assert characteristics of these classes and properties. We now present a discussion on the various constructs in DAML+OIL with their foundations in Description Logics (DLs)[9].

DAML+OIL is, in essence equivalent to a very expressive DL, with a DAML+OIL ontology corresponding to a DL terminology. As in a DL, DAML+OIL classes can be names (URIs) or *expressions*. A variety of constructors (or operators) are provided for building class expressions. The expressive power of the language is determined by the class (and property) constructors provided, and by the kinds of axioms allowed. **Table 1** summarizes the constructors used in DAML+OIL expressed using the standard DL syntax. In the RDF syntax, the expression `Bacterium ∩ Virus` would be written as:

```
<daml:Class>
        <daml:intersectionOf
          rdf:parseType="daml:collection">
            <daml:Class
                    rdf:about="#Bacterium"/>
            <daml:Class rdf:about="#Virus"/>
        </daml:intersectionOf>
</daml:Class>
```

The meanings of the first three constructors from **Table 1** are just the standard boolean operators on classes. The *oneOf* constructor allows classes to be defined by enumerating their members. The *toClass* and *hasClass* constructors correspond to slot constraints in a frame-based language.

Table 1: DAML+OIL class constructors

Constructor	DL Syntax	Example
intersectionOf	$C_1 \cap \ldots \cap C_n$	`Bacterium ∩ Animal`
unionOf	$C_1 \cup \ldots \cup C_n$	`Bacterium ∪ Virus`
complementOf	$\neg C$	`¬Plant`
oneOf	$\{x_1,\ldots, x_n\}$	`{aspirin, tylenol}`
toClass	$\forall P.C$	`∀partOf.Cell`
hasClass	$\exists P.C$	`∃processOf.Organism`
hasValue	$\exists P.\{x\}$	`∃treatedBy{aspirin}`
minCardinalityQ	$\geq n\ P.C$	`≥ 2 hasPart.Cell`
maxCardinalityQ	$\leq n\ P.C$	`≤ 1 hasPart.Tissue`
cardinalityQ	$= n\ P.C$	`= 1 partOf.Cell`

The class $\forall P.C$ is the class, all of whose instances are related via the property P only to resources of type C, while the class $\exists P.C$ is the class, all of whose instances are related via the property P to at least one resource of type C. The *hasValue* constructor is just shorthand for a combination of *hasClass* and *oneOf*. The *minCardinalityQ*, *maxCardinalityQ* and *cardinalityQ* constructors (known in DLs as qualified number restrictions) are generalizations of the *hasClass* and *hasValue* constructors. The class $\geq n\ P.C$ ($\leq n\ P.C$, $= n\ P.C$) is the class all of whose instances are related via the property P to at least (at most, exactly) n *different* resources of type C. The emphasis on different is because there is no unique name assumption *wrt* to resource names (URIs) and it is possible that many URIs could name the same resource.

Table 2 (next page, bottom) summarizes the axioms allowed in DAML+OIL. These axioms make it possible to assert *subsumption* or *equivalence wrt* classes or properties, the *disjointness* of classes, the *equivalence* or *non-equivalence* of individuals (resources), and various properties of properties. A crucial feature of DAML+OIL is that *subClassOf* and *sameClassAs* axioms can be applied to arbitrary class expressions. The last two rows of **Table 2** refer to DAML+OIL constructs *domain/range*, which identify the *domain* and *range* classes of the various properties. Their DL constructors are as shown. We shall discuss later in the paper, various approaches to represent domains and ranges and the impact it might have on the complexity of the reasoning process. DAML+OIL also allows properties of properties to be asserted. It is possible to assert that a property is unique (i.e., functional) and unambiguous (i.e., its inverse is functional). It is also possible to use inverse properties and assert that a property is transitive.

DAML+OIL REPRESENTATION OF THE SEMANTIC NETWORK

We now present a DAML+OIL representation of a small portion of the UMLS Semantic Network[2]. The Semantic Network types are represented using DAML+OIL A simplified version, after removing namespaces related markup of some of the Semantic Network types is presented below.

```
<daml:Class rdf:ID="Organism"/>
<daml:Class rdf:ID="Fungus"/>
<daml:Class rdf:ID="Virus"/>
<daml:Class rdf:ID="Bacterium"/>
...
```

Relationships in the Semantic Network are represented using the DAML+OIL object properties. It may be noted that many relationships in the Semantic Network are ***polymorphic***, i.e., they have multiple domains and ranges (e.g., `part_of`, `disrupts`) and will be discussed in the next section.

```
<daml:ObjectProperty rdf:ID="property_of">
    <rdfs:domain
       rdf:resource="#OrganismAttribute">
    <rdfs:range rdf:resource="#Organism">
</daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="process_of">
    <rdfs:domain
         rdf:resource="#BiologicFunction">
     <rdfs:range rdf:resource="#Organism">
</daml:ObjectProperty>
...
```

Axioms in the Semantic Network originate from the following sources.

- The type inheritance hierarchy.
- The property inheritance hierarchy.
- Inverse relationship constraints
- Rewriting of domain and range constraints.

The type hierarchy in the Semantic Network can be represented as a collection of subclass axioms. Some examples (in the DL syntax) are:

```
Fungus ⊆ Organism
Virus ⊆ Organism
Bacterium ⊆ Organism
Animal ⊆ Organism
Plant ⊆ Organism
...
```

The relationships in the Semantic Network also form a hierarchy, i.e., some relationships are sub-relationships of other relationships. This can be expressed using the *subPropertyOf* construct in DAML+OIL as illustrated below:

```
part_of ⊆ physically_related_to
contains ⊆ physically_related_to
```

```
property_of ⊆ conceptual_part_of
conceptual_part_of ⊆ conceptually_related_to
location_of ⊆ spatially_related_to
...
```

All relationships in the Semantic Network have inverse relationships defined for each other. This is represented using the inverseOf construct in DAML+OIL as illustrated below:

Asymmetric properties:

```
part_of ≡ has_part¯
evaluation_of ≡ has_evaluation¯
process_of ≡ has_process¯
```

Symmetric properties:

```
co-occurs_with ≡ co-occurs_with¯
adjacent_to ≡ adjacent_to¯
...
```

One strategy of handling multiple domains and ranges of properties (discussed later) is to use property restrictions to represent them by their DL equivalents (illustrated in **Table 2**). A rewriting for the relationship `property_of` is as follows:

```
T ⊆ ∀property_of.Organism (range constraint)
T ⊆ ∀has_property.OrganismAttribute (domain
constraint)
or ∃property_of.T ⊆ Organism
(in case the property_of¯ did not exist)
```

REQUIREMENTS SPECIFIC TO THE UMLS SEMANTIC NETWORK

The exercise of representing the Semantic Network using DAML+OIL constructs lead us to two areas where the preferred representation choice is not obvious, viz., representation of polymorphic relationships, and blocking inheritance of properties down some subclass links.

Polymorphic Relationships

Polymorphic relationships are relationships whose arguments, i.e., domain and range, can be instances of multiple classes, and the instances of domains and ranges have to be associated with each other. For example, consider a property P as follows:

$\text{domain}(P) = D_1$ and $\text{range}(P) = R_1$
$\text{domain}(P) = D_2$ and $\text{range}(P) = R_2$
where D_1, D_2, R_1, R_2 are classes that may be disjoint with each other s.t if $(x,y) \in P$, then:
either $x \in D_1, y \in R_1$ or $x \in D_2, y \in R_2$
but not $x \in D_1, y \in R_2$ or $x \in D_2, y \in R_1$
According to DAML+OIL Semantics[5], multiple domains and ranges are interpreted as intersections of their respective class expressions. In that case,
$\text{domain}(P) = D_1 \cap D_2$ and $\text{range}(P) = R_1 \cap R_2$
then, $x \in D_1 \cap \neg D_2$, $y \in R_1 \cap \neg R_2$ is an example of a *missed* model.

We now present different approaches to represent polymorphic relationships.

Domain/Range Factorization

This is a simple and special case of multiple domains and ranges, where each class in the domain is associated with each class in the range, i.e.
$\forall i\ \forall j$ $\text{domain}(P) = D_i$ and $\text{range}(P) = R_j$
In this case, the domain/range constraints can be specified as follows:
$\text{domain}(P) = D_1 \cup \ldots \cup D_m\ (1 \le i \le m)$
$\text{range}(P) = R_1 \cup \ldots \cup R_n\ (1 \le j \le n)$

Consider the relationship `analyzes`:

```
analyzes(DiagnosticProcedure, BodySubstance)
analyzes(LaboratoryProcedure, BodySubstance)
analyzes(DiagnosticProcedure, Chemical)
analyzes(LaboratoryProcedure, Chemical)
```

Table 2: DAML+OIL axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \subseteq C_2$	Human ⊆ Animal ∩ Biped
sameClassAs	$C_1 \equiv C_2$	Man ≡ Human ∩ Male
subPropertyOf	$P_1 \subseteq P_2$	part_of ⊆ physically_related_to
samePropertyAs	$P_1 \equiv P_2$	has_temperature ≡ has_fever
disjointWith	$C_1 \subseteq \neg C_2$	Vertebrate ⊆ ¬Invertebrate
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{heart_attack} ≡ {myocardial_infarction}
differentIndividualFrom	$\{x_1\} \subseteq \neg\{x_2\}$	{aspirin} ⊆ ¬{tylenol}
inverseOf	$P_1 \equiv P_2^{-}$	has_evaluation ≡ evaluation_of¯
transitiveProperty	$P^{+} \subseteq P$	part_of^{+} ⊆ part_of
uniqueProperty	$T \subseteq \le 1\ P$	T ⊆ ≤ 1 has_mother
unambiguousProperty	$T \subseteq \le 1\ P^{-}$	T ⊆ ≤ 1 is_mother_of¯
domain	$T \subseteq \forall P^{-}.C$ $\exists P.T \subseteq C$	T ⊆ ∀has_evaluation.Finding ∃evaluation_of.T ⊆ Finding
range	$T \subseteq \forall P.C$	T ⊆ ∀evaluation_of.OrganismAttribute

The domain/range constraints can be specified as:

```
domain(analyzes)
 = DiagnosticProcedure ∪ LaboratoryProcedure
range(analyzes) = BodySubstance ∪ Chemical
```

Property Renaming Approach

This approach involves renaming the property for each pair of domain and range classes specified and specifying *subPropertyOf* relationships. Consider a property P, s.t.

for $1 \leq i \leq n$, domain(P) = D_i and range(P) = R_i
For each i, create a property P_i, s.t.
  domain(P_i) = D_i and range(P) = R_i
  assert the constraint, $P_i \subseteq P$
assert $P \equiv P_1 \cup \ldots \cup P_n$

Consider the relationship `contains`:

```
contains(BodySpaceOrJunction,
                BodyPartOrganOrOrganComponent)
contains(BodySpaceOrJunction, BodySubstance)
contains(BodySpaceOrJunction, Tissue)
contains(EmbryonicStructure, BodySubstance)
contains(FullyFormedAnatomicalStructure,
                                 BodySubstance)
```

Renaming leads to the creation of new properties:

```
domain(contains_1) = BodySpaceOrJunction
range(contains_1=BodyPartOrganOrOrganComponent
contains_1 ⊆ contains
...
domain(contains_5)=
                FullyFormedAnatomicalStructure
range(contains_5) = BodySubstance
contains_5 ⊆ contains
```

Finally, the following constraint is asserted

```
contains ≡ contains_1 ∪ ... ∪ contains_5
```

Property Restrictions Approach

The final approach for expressing domain and range constraints, is for each class belonging to the domain of a property P, we assert a *toClass* property restriction on the class. Consider a property P, s.t.

domain(P) = D_1 and range(P) = R_1
domain(P) = D_2 and range(P) = R_2
The following axioms can be asserted:
$D_1 \subseteq \forall P.R_1$
$D_2 \subseteq \forall P.R_2$
For each concept $C \ni C \subseteq \neg (D_1 \cup D_2)$,
  assert the constraint: $C \subseteq \leq 0\ P$
The example discussed above can be represented as:

```
BodySpaceJunction ⊆
∀contains.(BodySubstance ∪ Tissue
            ∪ BodyPartOrganOrOrganComponent)
EmbryonicStructure ⊆ ∀contains.BodySubstance
FullyFormedAnatomicalStructure ⊆
                      ∀contains.BodySubstance
For each C ⊆
¬(BodySpaceOrJunction ∪ EmbryonicStructure ∪
              FullyFormedAnatomicalStructure)
    assert C ⊆ (≤ 0 contains)
```

This appears to be the most feasible of all the approaches discussed so far, though a comparative analysis of the complexities is required.

Blocking inheritance of Relationships

In some cases, we needed to block the inheritance of relationships to the subtypes of a semantic type to prevent nonsensical conclusions. The type in question might either be the domain or the range of a relationship.

Domain Blocking

The inheritance of a relationship is blocked for a subclass of a domain class. Consider the following example:

```
domain(process_of) = BiologicFunction
range(process_of) = Organism
```

If the relationship is inherited, we would have

```
domain(process_of) = MentalProcess
range(process_of) = Plant
```

A `Plant` is not a sentient being and cannot have a `MentalProcess`. Hence, we block the inheritance of the relationship `process_of` to `MentalProcess` by expressing the domain constraint as:

```
domain(process_of)
= BiologicFunction ∩ ¬MentalProcess
```

Alternatively, we can use property restrictions and rewriting of the domain constraints as follows:

```
MentalProcess ⊆ ≤ 0 process_of
```

Using qualified cardinality (maxCardinalityQ):

```
BiologicFunction ∩ ¬MentalProcess
   ⊆ ≤ 0 process_of Plant
```

Rewriting of the domain constraint gives:

```
∃process_of.T ⊆
         (BiologicFunction ∩ ¬MentalProcess)
```

Range Blocking

The inheritance of a relationship is blocked for a subclass of a range class. Consider the following example:

```
domain(conceptual_part_of) = BodySystem
range(conceptual_part_of)=
               FullyFormedAnatomicalStructure
```

If the relationship is inherited, we would have

```
domain(conceptual_part_of) = BodySystem
range(conceptual_part_of) = Cell
```

A `BodySystem` cannot be a part of `Cell`. Hence, we block the inheritance of the relationship `conceptual_part_of` to `Cell` by :

```
range(conceptual_part_of)
    = FullyFormedAnatomicalStructure ∩ ¬Cell
```

Alternatively, we can use property restrictions and rewriting of the range constraints as follows:

$$\mathtt{Cell} \subseteq\ \leq 0\ \mathtt{has\_conceptual\_part\ where}$$
$$\mathtt{has\_conceptual\_part} \equiv \mathtt{conceptual\_part\_of}^{-}$$

Using qualified cardinality (maxCardinalityQ):

$$\mathtt{BodySystem} \subseteq\ \leq 0\ \mathtt{conceptual\_part\_of}\ (\mathtt{FullyFormedAnatomicalStructure} \cap \neg\mathtt{Cell})$$

Rewriting the range constraint gives:

$$\mathtt{T} \subseteq \forall\mathtt{conceptual\_part\_of}.\ (\mathtt{FullyFormedAnatomicalStructure} \cap \neg\mathtt{Cell})$$

In general, Consider a domain (range) class D (R) with subclasses $D_1, \ldots, D_k$, $(R_1, \ldots, R_k)$, to which the property P needs to be inherited and subclasses $D_{k+1}, \ldots, D_n$ $(R_{k+1}, \ldots, R_n)$, for which it needs to be blocked. The above examples can be summarized as:

$\forall i,\ k+1 \leq i \leq n,\ \mathrm{domain}(P) = [D \cap \neg(\cup D_i)]$
$\forall i,\ k+1 \leq i \leq n,\ D_i \subseteq\ \leq 0\ P$ (using cardinality)
$\forall i,\ k+1 \leq i \leq n,\ [D \cap \neg(\cup D_i)] \subseteq\ \leq 0\ P\ R$ (qualified card)
$\forall i,\ k+1 \leq i \leq n,\ \exists P.T \subseteq [D \cap \neg(\cup D_i)]$ (definition)

$\forall i,\ k+1 \leq i \leq n,\ \mathrm{range}(P) = [R \cap \neg(\cup R_i)]$
$\forall i,\ k+1 \leq i \leq n,\ R_i \subseteq\ \leq 0\ P^{-}$ (using cardinality)
$\forall i,\ k+1 \leq i \leq n,\ D \subseteq\ \leq 0\ P\ [R \cap \neg(\cup R_i)]$ (qualified card)
$\forall i,\ k+1 \leq i \leq n,\ T \subseteq \forall P.[R \cap \neg(\cup R_i)]$ (definition)

CONCLUSIONS AND FUTURE WORK

We investigated the adequacy of the representational constructs in DAML+OIL for representing the knowledge in the Semantic Network. Though the DAML+OIL specification was adequate for our needs, there were multiple ways of representing the same knowledge. We investigated approaches for representing polymorphic relationships and identified two possible extensions to the DAML+OIL specifications:

- Support for operations such as union, intersection, etc. on properties (as illustrated in the property renaming approach). However this might lead to tractability problems.
- The ability to modify the meta-model. For example, the relationship *part_of* is a frequently occurring relationship in the biomedical domain, and there might be value in including it as a DAML+OIL construct with the same status as the *subClassOf* construct.

The main motivations for a *formal* representation of biomedical knowledge are: (a) creation and maintenance of *consistent* biomedical terminology; (b) enabling translations of concepts across multiple autonomous vocabularies; and (c) improved specification of queries for information retrieval. An instance of the latter is the annotation of MEDLINE documents using descriptors built with concepts from the MeSH vocabulary. For example, the semantics of the keyword “mumps” can be specified by the MeSH descriptor `(Mumps/CO AND Pancreatitis/ET)`. This semi-formal descriptor can be used to improve text retrieval by use as a label or as part of a query. It can also be expressed using a DL concept like $\exists\mathtt{complication.Mumps} \cap \exists\mathtt{etiology.Pancreatitis}$, enabling inferences during query answering.

These inferences can help recognize inconsistent (empty) concepts/relationships, and faulty subclass/ sub-property relationships for terminology creation and consistency management[6]. They also enable inference of concept equivalence for matching of search queries and document annotations. These inferences can also be used to merge vocabularies/ontologies into a directed acyclic graph (DAG) structure, given inter-vocabulary relationships[12]. Concept translations across vocabularies can then be determined by navigation in the merged graph[10].

Acknowledgements

Acknowledgements are due to Alexa McCray for enlightening discussions on the Semantic Network and Olivier Bodenreider and Patti Brennan for comments on the draft.

REFERENCES

1. Lindberg D, Humphreys B, McCray A. The Unified Medical Language System. Methods Inf Med 1993:32(4):281-91.
2. McCray A, Nelson S. The representation of meaning in the UMLS. Methods Inf Med 1995:34(1-2):193-201
3. Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American, May 2001. http://www.sciam.com/2001/0501issue/0501berners-lee.html
4. Resource Description Framework (RDF), http://www.w3.org/RDF
5. The DARPA Agent Markup Language. http://www.daml.org.
6. Stevens R, Goble C, Horrocks I and Bechhofer S. Building a Bioinformatics Ontology using OIL. IEEE Information Technology in Biomedicine (to appear), special issue on Bioinformatics
7. XML Schema, http://www.w3.org/XML/Schema
8. RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema
9. Horrocks I, Patel Schneider P F, van Hermelen F. An Ontology Language for the Semantic Web. Proceedings of the 18th National Conference on Artificial Intelligence (AAAI- 2002).
10. The Semantic Vocabulary Interoperation Project, http://cgsb2.nlm.nih.gov/~kashyap/projects/SVIP
11. McCray A, Srinivasan S, Browne A. Lexical methods for managing variation in biomedical terminologies. Proc Annu Symp Comput Appl Med Care 1994:235-9
12. Mena E, Kashyap V, Illarramendi A and Sheth A. Imprecise answers in a Distributed Environment: Estimation of Information Loss for Multiple Ontology-based Query Processing.” Int. J. of Cooperative Information Systems (IJCIS), 9(4), December 2000.