

# Classes versus Individuals: Fundamental Design Issues for Ontologies on the Biomedical Semantic Web

Matthias SAMWALD

*Section on Medical Expert and Knowledge-Based Systems, Core Unit for Medical Statistics and Informatics, Medical University of Vienna, Vienna, Austria*

**Abstract.** Most entities in biological discourse are in fact classes and not singular entities. This complicates the ontological description of biomedical reality in the Web Ontology Language (OWL), which is one of the fundamental standards of the Semantic Web. While the class—like nature of most biological entities seems to necessitate the use of OWL classes to describe biological reality in a consistent manner, this approach is not practically feasible for several reasons. I propose the use of OWL individuals together with special properties (the ‘CASA property pattern’) to overcome these problems.

Keywords: Medical Informatics, Bioinformatics, Internet, Semantic Web, OWL, RDF, Ontology.

## 1. Introduction

We can distinguish two general strategies for knowledge representation in biomedicine: the linguistic approach (which is focused on describing term meanings) and the realist ontological approach (which is focused on describing things in reality itself). The former approach is increasingly disregarded in favour of the latter, mainly because the ontological approach is more intuitive and self—consistent.

If one chooses to follow the ontological approach, it soon becomes obvious that biomedical research almost exclusively deals with *classes* of biological entities, not with individuals. The statement “*E. coli* contains the *LacZ* gene” normally does not refer to a certain *E. coli* cell in a certain petri dish, it refers to *E. coli*—cells in general.

This fact can be problematic when we want to employ a knowledge representation language like OWL DL, which is a core standard of the Semantic Web development [1]. OWL DL is based on description logics, which imposes strict restrictions on the ways in which classes and their instances (‘individuals’ in OWL terminology) can be used and related.

These restrictions do not pose a problem when an ontology commits to a certain system level of reality. For instance, the increasingly popular *Biological Pathways Exchange* ontology (BioPAX, [2]) employs a rather flat hierarchy of classes. Classes like ‘Protein’ or ‘Pathway’ as the most specialised classes in the ontology and are supposed to be directly instantiated. In one context, a molecular biologist might instantiate the class *Protein* with the individual ‘*Serotonin\_Receptor*’, which is a clear enough description in many cases. In another context, a neuroscientist might instantiate the class *Protein* with the individual ‘*Serotonin\_Receptor\_2A*’. *Serotonin\_Receptor\_2A* is in fact a subclass of

*Serotonin\_Receptor*, but this information cannot be represented in accordance to OWL DL semantics, since both have been defined as individuals.

The restrictions in the use of classes and individuals in OWL DL as compared to OWL Full are supposed to produce an advantageous trade-off between reasoning support and flexibility. However, in the case of biomedical ontologies on the Semantic Web, this trade-off seems less favourable. One of the main visions of the biomedical Semantic Web is the seamless integration of many different levels of description of biomedical reality. If one system level requires the use of instances, while another system level requires the use of classes, this integration is severely hampered by the restrictions of OWL DL.

The resulting ontologies are prone to become a tangled mixture of linguistic descriptions and realist ontological descriptions, which can easily lead to misconceptions, inconsistency and poor maintainability.

## 2. The first solution: If it is a class, then we *model* it as a class

A popular solution for the problems stated above is to rely solely on OWL classes for the representation of biological entities. The pragmatic notion of this approach is that everything that is a class in biological reality should also be modelled as a class in OWL. Examples of such ontologies are the OWL—version of Gene Ontology [3] or TAMBIS [4]. There have also been proposals for future versions of BioPAX that employ this solution. If we follow this solution, *Serotonin\_Receptor* would be defined as an OWL subclass of *Protein*, and *Serotonin\_Receptor\_2A* would be defined as a subclass of *Serotonin\_Receptor*.

Unfortunately, this approach suffers from several shortcomings:

**Problem 1)** Having a soft distinction between a ‘schema’ (classes) and ‘real data’ (individuals) can be a very beneficial thing in practice — the roles between ‘OWL developers’ and ‘OWL users’ are set. Making the creation of new classes mandatory for the representation of ‘real data’ dissolves this distinction. It also makes it much harder to write software to edit and/or view ontologies and data.

**Problem 2)** Simple OWL users don’t produce tasty pizzas [5]. Novices to the use of OWL often make mistakes in the definition of classes, because the behaviour of subsumption rules can sometimes be counter—intuitive. Even small errors can have cascading side—effects that compromise the consistency of a whole ontology. This problem can only partly be remedied by more intuitive tools — Protégé [6] is already really good, but still far from being something that can be let loose on the potential ‘OWL end—user’.

**Problem 3)** Making a class the value of a property of an individual is only possible in *OWL Full*, but not in *OWL DL*. Ontologies in *OWL Full* can only partially be handled by reasoners and many other software tools. Furthermore, these constructs make it difficult to constraint the range and allowed values of properties. This is a grave problem: In the biomedical domain, there are many cases where we want to relate a specific experiment, measurement or database entry (represented as an individual in OWL) to a biological entity (represented as a class if we follow solution 1). Several OWL DL – compatible workarounds for this problem have been proposed [7]. However, none of them is particularly elegant or intuitive.

**Problem 4)** While relations between individuals are represented as simple subject—predicate—object triples in the RDF data model, relations between OWL classes have to

be represented as *property restrictions*. Property restrictions are inefficiently mapped to the RDF triple—model, owing to the fact that OWL was designed on top of the RDF specification. As a result, the number of triples needed to represent information multiplies, leading to issues with memory and performance. All RDF stores that are scalable to large amounts of data (e.g. *Sesame* [8]) are optimized for the common RDF data model and do not perform optimizations for these special OWL constructs. Furthermore, the simple and elegant triple—based data model of RDF is obfuscated. The resulting graph becomes very hard to read.

For example, the statement “Protein\_A and Protein\_B are proteins, Protein\_A binds to Protein\_B” can be represented with OWL individuals and properties like this:

```
example:Protein_A  rdf:type          example:Protein
example:Protein_B  rdf:type          example:Protein
example:Protein_A  example:bindsTo  example:Protein_B
```

Even this simple statement becomes quite confusing when represented with OWL classes and property restrictions:

```
example:Protein_A  rdf:type          owl:Class
example:Protein_A  rdfs:subClassOf  example:Protein
_:bn1              rdf:type          owl:Restriction
_:bn1              owl:onProperty  example:bindsTo
_:bn1              owl:someValuesFrom example:Protein_B
example:Protein_A  rdfs:subClassOf  _:bn1
example:Protein_B  rdf:type          owl:Class
example:Protein_B  rdfs:subClassOf  example:Protein
```

Of course, this problem could be solved by the development of OWL stores that are optimized for the representation of these constructs, but such developments are very unlikely to happen in the near future. In fact, even most of the databases for simple RDF that are currently available are not efficient enough to handle the large quantities of data encountered in bioinformatics [9].

There is a certain time—window for the success of the Semantic Web technologies, so ontology developers should put an emphasis on producing ontologies that can actually work on systems that are available in the present and near future. Otherwise, we might see OWL being abandoned in favour of other solutions.

**Problem 5** Because the representation of property restrictions leads to obfuscated RDF graphs, RDF Query languages (like SPARQL [10]) can hardly be used to query data. The lack of a query language with built-in support for these special OWL constructs is a grave limitation.

**Problem 6** A minor limitation of the use of classes is that it offers no means of defining sub-properties of the ‘subclass of’ relation to state things like ‘phosphorylated\_Protein\_A – phosphorylated\_form\_of – Protein\_A’. In some scenarios, the possibility to define such sub-properties might be practical.

It can be concluded that the first solution is very consistent from an ontological viewpoint, but it is simply not feasible for real—world use on the Semantic Web. It obfuscates the

simplicity and elegance of the RDF data model, is not scalable and will hamper adaptation of RDF/OWL by developers and end—users.

### 3. The second solution: Bringing individuals back into play

Since the use of classes for the representation of biological reality has been shown to be formally correct but markedly impractical, it might become necessary to search for other ways of representation in OWL. I propose the use of special properties (termed *non — individualising properties*) to refer to individuals in a way that addresses their class—like character while keeping the ontology as consistent as possible. This design pattern has not been applied to real—world use cases as of yet, so the practicality of this approach still needs to be established.

#### 4.1 Non – individualising properties: the CASA property pattern.

CASA is an acronym for ‘can-always-some-all’, the prefixes and suffixes used to distinguish these special properties from other properties. The generic properties of this pattern are listed and described in Table 1.

Table 1: The five basic properties making up the CASA pattern.

<p><b>can_referenceTo_some</b>  <i>Meaning:</i> If A can_referenceTo_some B, then some entities in reality that ‘are an A’ stand in some relation to some entities in reality that ‘are a B’.  <i>Example:</i> Serotonin can_bindsTo_some SerotoninReceptor  <i>Sub-properties:</i> always_referencesTo_some, can_referenceTo_all</p>
<p><b>always_referencesTo_some</b>  <i>Meaning:</i> If A always_referencesTo_some B, then all entities in reality that ‘are an A’ stand in some relation to some entities in reality that ‘are a B’.  <i>Example:</i> LivingCell always_contains_some DNA  <i>Sub-properties:</i> always_referencesTo_all, always_is_some</p>
<p><b>can_referenceTo_all</b>  <i>Meaning:</i> If A can_referenceTo_all B, then some entities in reality that ‘are an A’ stand in some relation to all entities in reality that ‘are a B’.  <i>Example:</i> DNA can_beContainedIn_some LivingCell  <i>Sub-property:</i> always_referencesTo_all</p>
<p><b>always_referencesTo_all</b>  <i>Meaning:</i> If A always_referencesTo_all B, then all entities in reality that ‘are an A’ stand in some relation to all entities in reality that ‘are a B’. This property can also be used to relate entities that are ‘true’ individuals in reality.  <i>Examples:</i> Death always_endOf_all Life,  TajMahal always_locatedIn_all India</p>
<p><b>always_is_some</b>  <i>Meaning:</i> acts like the common ‘is_a’ or ‘subclassOf’ – relation known from many ontology frameworks. ‘always_is_some’ is transitive and should be used with the same rules in mind that apply to the definition of subclasses. This means that if A always_is_some B, and B always_is_some C, then it should hold true that A always_is_some C.  <i>Example:</i> phosphorylated_MAPKinase always_is_some MAPKinase</p>

The first four properties listed in Table 1 are not meant to be used directly; they just act as a generic scaffold. An ontology can define sub—properties of these generic properties that have the same prefixes and suffixes. For example, a biomedical ontology might define a property ‘*can\_bind\_some*’ to represent protein binding. Figure 1 shows an example of a possible property hierarchy that can be derived according to this design pattern.

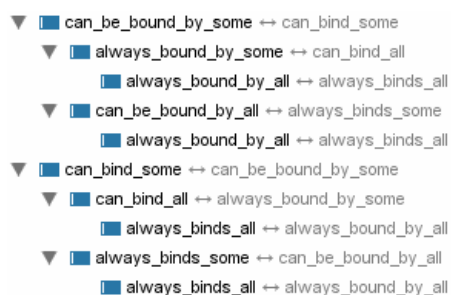


Figure 1: The CASA property pattern demonstrated with the property *binds* and its inverse property *bound\_by*. Sub-properties are indented. The relations between each property and its inverse property are given. Screenshot from Protégé 3.2 [6].

Of course, it is not necessary to define the whole hierarchy with prefixes and suffixes for every property in an ontology. In biomedical ontologies, most of the possible relations between entities are not universally true, so it might suffice to solely define properties of the *can\_referenceTo\_some* – type.

According to this design pattern, every resource can be a ‘real’ individual, a class, or both. Its identity as a class or as an individual is defined solely through its context, i.e. the type of properties that point to and from the resource.

The proposed hierarchy of sub-properties insures that a possible transitivity of properties is handled correctly. For example, we can define the transitive property *can\_bePartOf\_some* and its transitive sub-property *always\_partOf\_all*. Consequently, if we make the three statements

A	<i>always_partOf_all</i>	B
B	<i>can_bePartOf_some</i>	C
C	<i>always_partOf_all</i>	D

it follows that A *can\_bePartOf\_some* D. It does not follow, however, that A *always\_partOf\_all* D, because the chain of ‘stricter’ always/all—relations has been broken by the ‘softer’ can/some—relation.

The *always\_referencesTo\_some* and the *can\_referenceTo\_all* properties should not be declared symmetric. Instead, *can\_referenceTo\_some* or *always\_referencesTo\_some* should be used for symmetric properties.

## 5. Conclusion

Compared to the first solution, the proposed second solution offers a better balance between ontological coherence and capability of fully automated reasoning on one side, and simplicity, scalability, flexibility and error tolerance on the other side.

One drawback of this design pattern is that it gives ontology designers the ability to model any concept either as a class or as an instance. This adds a further degree of freedom to the design process, which may result in even more diverse ways of representing identical facts; thereby decreasing the consistency between different ontologies. It is therefore necessary to develop recommendations on when things should be modelled as classes or as individuals. In light of the arguments above, it might be prudent to put the emphasis on the use of individuals. The current BioPAX ontology might be a good example for such a design philosophy.

Another drawback of this solution is that while these special properties are understandable to humans, they do not adhere to the standard semantics of OWL. This means that there is no dedicated support for these constructs in current OWL reasoners.

The proposed design pattern is just one of many approaches that try to ease the integration of different system levels and granularities encountered in biomedical reality. In this case, I addressed practical issues in the representation of specialisation and abstraction. Other proposals address the integration of different scales and collectivity [11]. The development and acceptance of such design patterns is a necessity for the further development of the biomedical Semantic Web.

## References

- [1] <http://www.w3.org/TR/owl-features/> (last accessed Mar. 11, 2006)
- [2] <http://www.biopax.org/> (last accessed Mar. 11, 2006)
- [3] <http://www.geneontology.org/> (last accessed Mar. 11, 2006)
- [4] <http://imgproj.cs.man.ac.uk/tambis/details.html> (last accessed Mar. 11, 2006)
- [5] Rector A, Drummond N, Horridge M, Rogers J, Knublauch H, Stevens R et al. OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns. 3257 ed. 2004.
- [6] <http://protege.stanford.edu/>.
- [7] Representing Classes As Property Values on the Semantic Web (W3C Working Draft). <http://www.w3.org/TR/2005/NOTE-swbp-classes-as-values-20050405/> (last accessed Mar. 11, 2006)
- [8] Broekstra J, Kampman A, van Harmelen F. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. 2342 ed. 2002.
- [9] Lee R. Scalability Report on Triple Store Applications. Technical Report, SIMILE project . 2004.
- [10] <http://www.w3.org/TR/rdf-sparql-query/> (last accessed Mar. 11, 2006)
- [11] Rector A, Rogers J. Granularity Scale & Collectivity: When size does and doesn't matter (unpublished).