3

Technologies and best practices for building bio-ontologies

Mikel Egaña Aranguren, Robert Stevens, Erick Antezana, Jesualdo Tomás Fernández-Breis, Martin Kuiper, and Vladimir Mironov

3.1 Introduction

Genomics technologies generate vast amounts of data of a wide variety of types and complexities, and at a growing pace. The analysis of such data and the mining of the resulting information is insufficient without a contextual interpretation, that is, biological knowledge deduced from the data. This knowledge states the data's biological meaning in terms of, for instance, molecular function, cellular location, or network interactions. Biological knowledge is diverse, vast, complex, and volatile. These factors, together with the nature of evolved systems, make the knowledge generated by the life sciences difficult to capture. As molecular biology has relatively recently included a systems approach, it has become increasingly important to have precise and rich representations of the catalogs that in turn form the basis of the networks and pathways that describe biological systems. Therefore, biological knowledge management is becoming essential for current research in life sciences (Antezana *et al*., 2009a).

Biological knowledge has traditionally been represented in human interpretable formats like natural language in scientific literature, or somewhat more

Knowledge-Based Bioinformatics: From Analysis to Interpretation Edited by Gil Alterovitz and Marco Ramoni

structured in database entries. The heterogeneous terminology used, together with the natural language form, has made it difficult to manage and use that knowledge, for both humans and, more importantly, computers. In order to use the computers' ability to handle complex and large amounts of information, it has become clear that biological knowledge should be codified in a machine interpretable form. Only in this way can biologists begin to exploit their hard-won data.

A widely used method for codifying knowledge in a machine interpretable form is to represent it in ontologies. Ontologies are computational formalizations of the concepts shared by a community of scientists. Thus, ontologies can be used to describe and define the entities of a domain, and their relations, axiomatically, with precise semantics. The expression of knowledge with precise semantics makes it possible for computers to perform, via automated reasoning, information management tasks that can save scarce human resources and retrieve more complete results from biological knowledge (e.g., new hypotheses).

Therefore, the use of bio-ontologies, that is, ontologies that represent biological knowledge, is essential in biological knowledge management and integration, and they have become mainstream within bioinformatics. Currently, there are established communities of bio-ontologists, like the Open Biomedical Ontologies (OBO) Foundry (Smith *et al*., 2007; www.obofoundry.org/), which have produced important bio-ontologies such as the Gene Ontology (GO; Gene Ontology Consortium, 2000).

Many bio-ontologies exploit the very technology that will be used for building the Semantic Web (www.w3.org/standards/semanticweb/), which is the next 'smart' generation of the current Web, based on the automatic management of Web content. The W3C (www.w3.org/), the consortium responsible for the implantation of the Semantic Web and other open Web standards, has been fostering the Semantic Web Health Care and Life Sciences (HCLS) Interest Group (www.w3.org/blog/hcls) for working towards a Life Sciences Semantic Web (LSSW).

This chapter provides an introduction to the process of building bio-ontologies, analyzing the benefits and problems of modeling biological knowledge axiomatically, especially with regards to automated reasoning. Thus, the aspects that a biologist should consider in order to create a reusable, robust, rigorous, and axiomatically rich bio-ontology are briefly reviewed, providing pointers to successful engineering techniques and bio-ontologies. The aim of this chapter is not to provide a detailed methodology of the creation of bio-ontologies (the literature on the subject is vast); rather, the chapter highlights the elements that have to be taken into account, to help the reader to make informed decisions while building bio-ontologies.

3.2 Knowledge representation languages and tools for building bio-ontologies

An ontology represents knowledge through axioms. Axioms are used to describe the objects from the knowledge domain: their categories and the relationships between them. The axioms are written using a logical formalism, a Knowledge

Q1

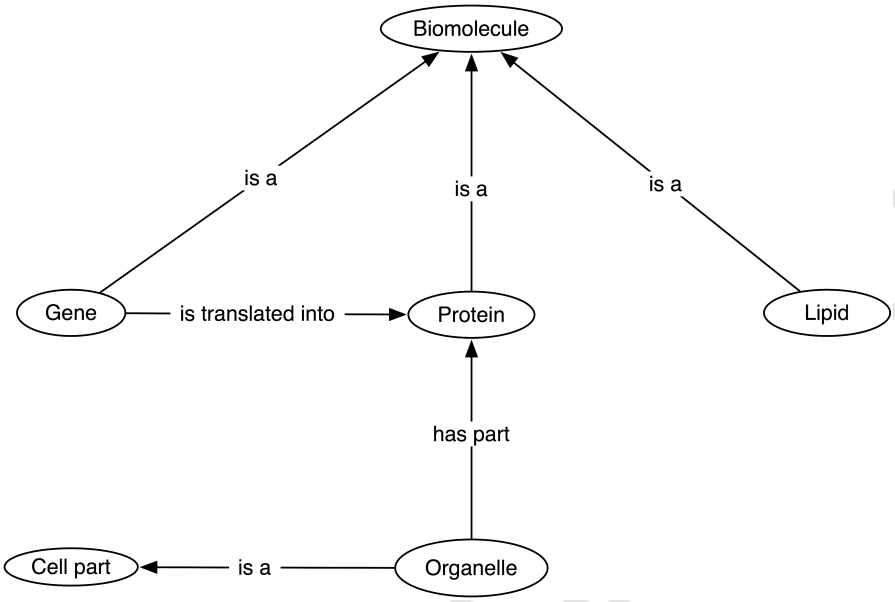

Figure 3.1 Simple bio-ontology, representing a 'toy' knowledge domain. The names of concepts – representing the categories or classes of objects in the domain – (e.g., `Protein`*) and relations among the objects (e.g.,* `has_part`*) are irrelevant for a computer; they are only 'understood' by humans. However, the structure of the ontology, expressed using axioms, is what the computer is able to manage, exploiting automated reasoning.*

Representation (KR) language, which enables their computational interpretation (Figure 3.1).

The semantics of a KR language defines the computational interpretation of the statements (axioms) the ontologist makes in an ontology, thus, how the computer 'understands' such statements. The different KR languages offer different levels of expressivity (what can be said about a domain); therefore, ontologists are able to make statements at different complexity levels, depending on the expressivity of the language of choice. Expressivity is related to computational tractability: the more expressive a language, the less tractable; that is, the more computational resources are needed by a computer to operate on an ontology written in such a language.

Currently, the most used KR languages in life sciences are the Resource Description Framework (RDF)[1], the Web Ontology Language (OWL)[2], and the

[1] RDF is not strictly a language for creating ontologies. However, using a broad definition of ontology, and considering the widespread use of RDF in the LSSW and its close relation to OWL, it has been included in this chapter.

[2] RDF Schema (RDFS) offers functionality close to OWL. However, it has been left out of this review due to the fact that it is not widely used in the LSSW, and for the sake of brevity.

Table 3.1 Summary of features of RDF, OWL, and OBO. SW, Semantic Web oriented (Semantic Web stack of protocols, URIs, etc.); LS, Life Sciences; LSHC (IG/KB), W3C Life Sciences and Health Care (Interest Group/Knowledge Base).

	Repositories	SW	Reasoning	Strong points	Weak points	Editors	APIs	Communities of practice	Outstanding projects
RDF	OBO foundry, Bioportal[a]	Yes	No	Widely used outside LS	Only triple like information	Protégé 3	Jena	LSHC IG	Bio2RDF, Biogateway, LSHC KB
				Simple and intuitive				All the SW communities	
				SPARQL					
OWL	OBO foundry, Bioportal	Yes	Yes	Widely used outside LS	Difficult to understand and use	Protégé 3, 4	OWL API	LSHC IG	OBI, BioPAX, CCO, PhosphaBase
				Expressive yet tractable		TopBraid composer[b]		All the SW communities	
OBO	OBO foundry, Bioportal, Ontology Lookup Service[c]	No	No	Widely used within LS	No formal semantics	OBO-Edit	ONTO-Perl	OBO foundry	GO, CL
				Simple and intuitive	Not used outside LS	COBrA-CT[d]			

[a] www.bioportal.bioontology.org/.
[b] www.topbraidcomposer.com/.
[c] www.ebi.ac.uk/ontology-lookup/.
[d] www.aiai.ed.ac.uk/project/cobra-ct/.

OBO format. They mainly differ in terms of expressivity, tool availability, and communities of practice. Since RDF and OWL are official W3C recommendations to implement the Semantic Web, they are also used outside the life sciences domain, whereas the OBO Format is only used to represent life-sciences-related information. As RDF and OWL are part of the Semantic Web stack of technologies, OWL 'includes' RDF, and therefore an OWL ontology can be accessed with OWL-specific tools (OWL expressivity level) or RDF tools (RDF expressivity level). The following subsections describe the main features of each language, as summarized in Table 3.1.

3.2.1 RDF (resource description framework)

RDF (www.w3.org/TR/rdf-primer/) was designed to represent information about Web resources in the Semantic Web, thus to publish data in a basic machine processable form. The information in RDF is represented in statements formed by a subject, a predicate and an object, called triples. For example, a triple in RDF would read `SWI4 participates_in G1/S_transition.` `SWI4` is the subject, `participates_in` the predicate, and `G1/S transition` the object (Figure 3.2). Triples can be combined to form a graph (Figure 3.3). In an RDF graph, the subject of a triple can be the object of another triple.

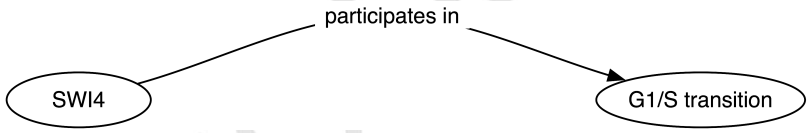

Figure 3.2 An RDF triple. A subject (`SWI4`) is related to an object (`G1/S_transition`) by a predicate (`participates_in`).

RDF uses URIs (Uniform Resource Identifiers; www.w3.org/standards/techs/uri) to identify entities (subjects, predicates, and objects). The use of URIs provides the possibility of referring to entities from different graphs that have been published in different resources on the Web. This enables a framework to combine graphs from different resources, or to combine graphs at query time.

RDF graphs can be queried using SPARQL (www.w3.org/TR/rdf-sparql-query/). SPARQL is a query language that can be used to retrieve smaller graphs from a target graph. In order to perform the retrieval, a user must define a query graph in which one or more entities are left as variables, and the query graph is matched against the target graph, returning the appropriate answer as a smaller sub-graph of the target graph.

RDF is based on a simple model that enables the representation of diverse information with very low computational costs, provided that such information can be captured as a set of subject–predicate–object triples. Therefore, the manipulation of RDF graphs through APIs (Application Programming Interfaces) like

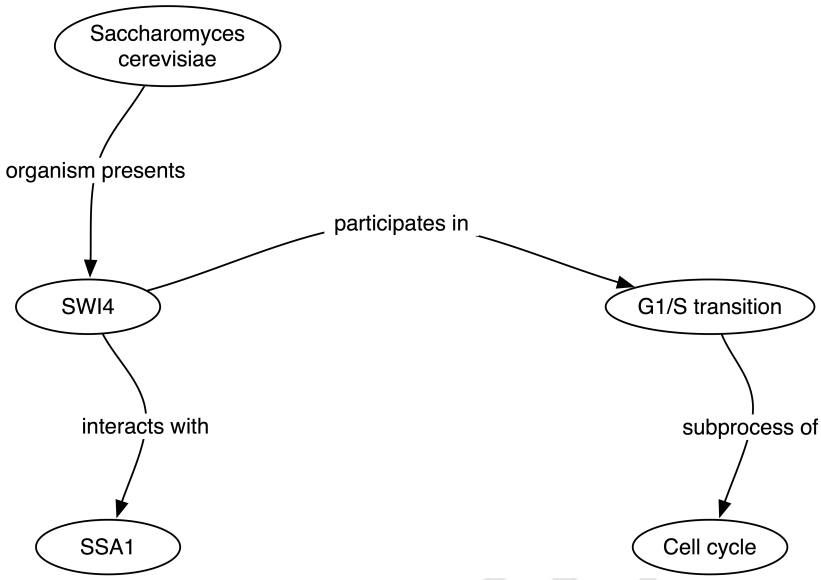

Figure 3.3 An RDF graph made by combining four triples. The triples share some common entities, such as `SWI4`*, which is the subject of two triples (*`participates_in G1/S_transition` *and* `interacts_with SSA1`*) and the object of another triple (*`Saccharomyces_cerevisiae organism_presents`*).*

Jena (http://jena.sourceforge.net/) is straightforward. This simplicity has made RDF the chosen language in several bioinformatics resources such as BioGateway (www.semantic-systems-biology.org/biogateway), Bio2RDF (http://bio2rdf.org/), and HCLS KB (www.w3.org/TR/hcls-kb/).

3.2.2 OWL (Web ontology language)

OWL (www.w3.org/TR/owl2-overview/) was designed as a language to publish machine processable and interoperable ontologies in the Web. OWL, compared to RDF, offers a semantic vocabulary to describe a knowledge domain. Such expressivity may have a higher computational cost. Nevertheless, OWL allows the representation of biological information with a finer granularity, opening up ample possibilities for interesting applications such as automated reasoning.

The OWL semantics is based on three elements: individuals, classes (sets of individuals), and properties (two individuals, or an individual and a data value, are linked in a pair along a property; Figure 3.4)[3]. Classes are built by specifying

[3] An OWL ontology that has classes, individuals and properties can be considered a Knowledge Base (KB). If there are no individuals, the artifact can be considered simply an ontology. An ontology describes a schema with which some entities of the domain (individuals) are described; a KB includes the schema (ontology) and the individuals.

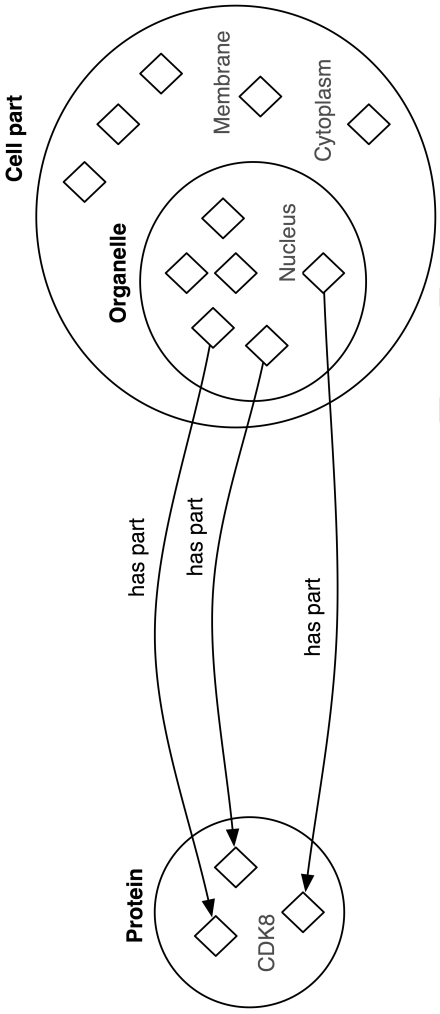

Figure 3.4 OWL classes (`Protein`, `Organelle`, `Cell_part`), individuals (`Nucleus`, `CDK8`, `Membrane`, `Cytoplasm`, and the rest of the diamonds), and an object property (`has_part`), used three times (in three pairs of individuals). The class `Organelle`, which has `Nucleus` as an individual (amongst others), is a subclass of `Cell_part`, which has other individuals (`Membrane`, `Cytoplasm`): every individual of `Organelle` is also an individual of `Cell_part`.

the conditions that the individuals should fulfill to belong to the class, in terms of which and how many relationships they should have, using class expressions. OWL offers universal (`only`) or existential (`some`) qualifiers and a plethora of typical logical constructs, such as negation (`not`), other Boolean operators (`or`, `and`), and more constructs, to create class expressions. Such constructs can be combined in complex (rich) class expressions. Class conditions can be either necessary (e.g., every nucleus is part of a cell, but being part of a cell is not enough to flag an organelle as nucleus) or necessary and sufficient (e.g., having a nucleolus as a part is a necessary and sufficient condition to flag an organelle as nucleus, as nuclei are the only organelles with nucleoli). The classes with at least one necessary and sufficient condition are called defined classes, whereas the classes with only necessary conditions are called primitive classes.

Classes can be subclasses of other classes, thus creating a taxonomy. The semantics of the subclass relation reads that, given a superclass S, every individual I of a given subclass of S is also an individual of S; for instance, all the organelles are cell parts, but not all the cell parts are organelles (membrane and cytoplasm are cell parts but are not organelles), therefore `Organelle` is a subclass of `Cell_part` (instead of an equivalent class).

There are three types of properties in OWL: properties that link pairs of individuals (object properties), properties that link individuals with data values (data type properties), and properties that can be used to add natural language information to axioms and entities, without affecting automated reasoning (annotation properties). Object properties can be arranged in hierarchies, and features of properties (such as transitivity) can be defined.

OWL can be expressed in various syntaxes. The most common computer readable syntax is RDF/XML (Figure 3.5). The Manchester OWL Syntax (MOS) offers a human-readable OWL syntax (Horridge *et al*., 2006). For example, the expression from Figure 3.5 would read as follows in MOS: `Nucleus subClassOf has_part some Protein`.

```
<owl:Class rdf:about="#Nucleus">
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resources="#has_part"/>
    <owl:someValuesFrom rdf:resource="#Protein"/>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Figure 3.5 OWL RDF/XML syntax of the MOS expression `Nucleus subClassOf has_part some Protein`.

OWL is based on Description Logics (DLs; Baader *et al*., 2003), a well known logical formalism. OWL offers an optimal balance between expressivity and tractability, allowing the efficient application of automated reasoning on OWL ontologies. Automated reasoning consists of using a program to infer axioms

from the axioms asserted in the ontology. The asserted axioms entail the inferred axioms. Thus, an automated reasoner makes axioms that were implicit explicit, showing further information to the bio-ontologist. For instance, let us consider the following two classes as being entities of our bio-ontology:

- `Nucleus`, with the axiom `Nucleus subClassOf part_of some Cell`. In order to be a nucleus it is necessary to be part of a cell, but being part of a cell is not enough on its own to be a nucleus (there are other organelles that are also part of a cell, but are not nuclei). Therefore, `Nucleus` is a primitive class.
- `Organelle`, with the axiom `Organelle equivalentTo part_of some Cell`. Anything that is part of a cell is an organelle. Therefore, `Organelle` is a defined class.

An automated reasoner will infer that `Nucleus` is a type of `Organelle`, thus the axiom `Nucleus subClassOf Organelle` will be made explicit or 'added' into the bio-ontology by the automated reasoner[4]. This is so because the axioms `Nucleus subClassOf part_of some Cell` and `Organelle equivalentTo part_of some Cell` entail the axiom `Nucleus subClassOf Organelle` (if all nuclei are part of a cell, and anything that is a part of a cell is an organelle, then nuclei are organelles).

The outcome of an automated reasoning process depends strongly on the axiomatic richness of the bio-ontology. It should also be noted that an automated reasoner acts in a 'ruthless' manner, showing the axioms that our modeling entails; in the above reasoning example, plasma membrane and cytoplasm should not be classified as organelles, indicating a likely modeling error on our side. It is necessary to regularly run an automated reasoner while building a bio-ontology, either to be reminded that our modeling is wrong or to highlight new information that was implicit ('hidden') in our modeling, entailed by the asserted axioms[5]. The more axioms we express in an ontology, the better; it is better to be axiomatically wrong (the automated reasoner tells us why we are wrong) than axiomatically correct and conceptually wrong (because we have not added those axioms). The automated reasoner shows the contradictions in our conceptual world.

In more concrete terms, automated reasoning can be used in the following ways:

(1) Perform complex queries against the knowledge stored in the ontology.

(2) Infer the class–subclass relationships from the class expressions; that is, build automatically the class hierarchy (taxonomy). For example, the

[4] This modeling (incorrectly) assumes that plasma membrane and cytoplasm should be classified as organelles; simplified for the sake of the example clarity.

[5] The automated reasoner will infer all the information entailed by the asserted axioms, including the information that a human would miss because of the extent or complexity of such information. That is why, among other reasons, automated reasoners can be so useful in knowledge-intensive disciplines like life sciences.

Normalization technique allows one to maintain a multiple inheritance in an ontology relying solely on the automated reasoner, provided that the appropriate class expressions are added to the ontology. In another example, an automated reasoner was used to check the completeness of the GO class hierarchy, in the Gene Ontology Next Generation (GONG) project (Egaña Aranguren *et al*., 2008a).

(3) Given an individual and its relationships to other individuals, the automated reasoner can infer to which class(es) it belongs.

(4) Check the consistency of the asserted axioms, as the automated reasoner can flag contradictory axioms. Such a procedure is used, for example, to ensure that the information gathered from different resources commits to the same schema (Miñarro-Gimenez *et al*., 2009).

Some OWL features stand out, apart from its expressivity, in terms of information integration:

- OWL (as well as RDF) relies on URIs to identify entities. Therefore, the Web machinery is also available for OWL.
- OWL is self-descriptive, that is, the schema and the data described using such schema are expressed in the same language: schema reconciliation is not needed, and the reconciliation problem is shifted to a more abstract (conceptual) level.
- Open World Assumption (OWA): OWL semantics interpret the absence of information as unknown rather than false. OWL assumes that, as the knowledge of the world we have is by definition incomplete, we cannot infer negation from the absence of information. Therefore, new information can be added to our bio-ontology and prior inferences remain valid, for example when importing entities from another OWL ontology. (However, a new inconsistency may be triggered.) This model fits with the biological knowledge domain, always being extended by different agents.
- Lack of Unique Name Assumption (UNA): in OWL, the fact that two entities have different names does not mean that they are different. Such entities need to be explicitly asserted to be different with the axioms `differentFrom` and `disjointWith`. On the other hand, different entities can also be asserted to be the same entity with the axioms `sameAs` and `equivalentTo`. For example, an OWL ontology can describe a gene with the name `CYC8`, and the same gene can be described in another OWL ontology with the name `SSN6`: they can be asserted to be the same entity (e.g., `CYC8 sameAs SSN6`), easing integration as no mapping must be created.

The expressivity and integrative features that OWL provides enable the representation of a considerable amount of biological concepts in a computationally accessible manner (Stevens *et al*., 2007). Such features have

promoted the use of OWL in several domains, and many tools supporting it have been also developed (www.w3.org/2007/OWL/wiki/Implementations), amongst which Protégé (http://protege.stanford.edu/) stands out as the most used OWL editor. Moreover, there are automated reasoners available for OWL, like Pellet (http://clarkparsia.com/pellet/) or FaCT++ (http://code.google.com/p/factplusplus/), and APIs like the OWL API (www.owlapi.sourceforge.net/). OWL has been successfully employed in projects such as OBI (www.purl.obolibrary.org/obo/obi), CCO (www.cellcycleontology.org/), BioPAX (www.biopax.org/), and PhosphaBase (www.bioinf.manchester.ac.uk/phosphabase/).

3.2.3 OBO format

The OBO format (www.geneontology.org/GO.format.shtml) has become the *de facto* KR language to model biological concepts for most of the OBO bio-ontologies, which are the most widely used bio-ontologies. Its development has been mainly fostered by the GO consortium (www.geneontology.org/). Figure 3.6 shows a sample entry of a term from the GO.

```
[Term]
id: GO:0005634
name: nucleus
def: "A membrane-bounded organelle of eukaryotic cells in which chromosomes
are housed and replicated. In most cells, the nucleus contains all of the cell's
chromosomes except the organellar chromosomes, and is the site of RNA synthesis
and processing. In some species, or in specialized cell types, RNA metabolism or
DNA replication may be absent." [GOG:go_curators]
synonym: "cell nucleus" EXACT []
xref: Wikipedia:Cell_nucleus
is_a: GO:0043231 ! intracellular membrane-bounded organelle
```

Figure 3.6 An OBO entry describing the term `Nucleus` *from the GO.*

In contrast to languages such as OWL, OBO has been tailored to the needs of the bio-ontologists (e.g., OBO offers an efficient mechanism for fine-grained annotations on ontology terms), resulting in the perception that it is more intuitive and more appropriate for biological knowledge modeling. Although OBO does not rely on any formal semantics, OBO algorithmic processing tools have been implemented, like the OBO-Edit reasoner (www.oboedit.org/docs/html/The_OBO_Edit_Reasoner.htm), the OBO Language (OBOL; Mungall, 2004), and the OBD-SQL reasoner (Mungall *et al*., 2010). OBO ontologies can also be translated into OWL to exploit automated reasoning, but such translation is not completely free of problems (Golbreich *et al*., 2007). In terms of expressivity, OBO can be used to represent relatively complex axioms, but composite expressions like `Nucleus subClassOf (part_of some Cell) and (has_part only (Nucleus_membrane or Nucleolus and not Ribosome)` cannot be expressed.

OBO is relatively human readable and easy to manipulate programmatically, with APIs like ONTO-PERL (Antezana *et al*., 2008), or graphically, with ontology editors like OBO-Edit (http://oboedit.org/). OBO has been successfully employed in very influential projects such as the GO or the Cell Type Ontology (CL; Bard *et al*., 2005). The GO is used for annotation by many current bioinformatics resources (www.ebi.ac.uk/GOA/). The CL is used in projects like XSPAN (www.xspan.org/).

3.3 Best practices for building bio-ontologies

Ontology building is still in a transition state from a 'craft' to a fully industrial engineering discipline (Bodenreider and Stevens, 2006). Therefore, there are neither established methodologies nor fully accepted principles. There are, however, practices that have already demonstrated their utility, and they are agreed to be important by the bio-ontologist community, explained as follows. Figure 3.7 summarizes such practices and the place they occupy in the bio-ontology development process.

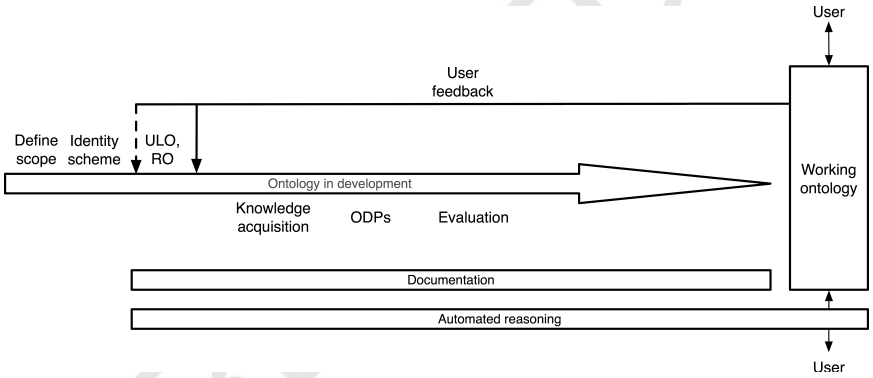

Figure 3.7 Diagram of the development cycle of a bio-ontology, with the best practices described in Section 3.3. The bio-ontology development starts by defining the scope, and it is repeated as necessary. User feedback is used to improve the bio-ontology, but generally without changing the scope and identity scheme, and barely changing the used ULO or set of relations. Documentation should be provided through the whole process. Automated reasoning should be used at development time (e.g., for consistency checking) and also users can exploit automated reasoning to query the ontology. Users can also interact with the ontology without using automated reasoning.

3.3.1 Define the scope of the bio-ontology

Bio-ontologies are able to perform a whole range of functions (Stevens and Lord, 2008). The function(s) of a bio-ontology will determine its scope and

'shape.' Therefore, explicitly and clearly defining the function (and hence the scope) of an ontology in early development stages, and sticking to such definition, is important to avoid spending too much effort in extending the ontology endlessly.

3.3.2 Identity of the represented entities

One of the most important elements of a LSSW is the identity of entities that form the biological knowledge domain, such as genes and proteins (Good and Wilkinson, 2006). Thus, many current bioinformatics resources describe the same entity with different identities (e.g., many resources give different names to the same gene). Different global identity schemes have been proposed to address the problem, but none has prevailed. The latest proposal is the Shared Names endeavour (http://sharedname.org/).

It is important to use an explicit identity scheme for the bio-ontology being built, and be consistent in its application. It might be that the identity scheme chosen does not 'succeed' and be used in the future by other resources, but nonetheless it will facilitate internal knowledge management, and if another identity scheme succeeds later on, it will be possible to map to it.

3.3.3 Commit to agreed ontological principles

There are ontological principles that are useful in order to make the bio-ontology interoperable with other bio-ontologies and resources. Such principles, however, impose a certain structure on our bio-ontology, and they determine strongly the subsequent modeling (Schulz *et al*., 2008). Therefore, the bio-ontologist must maintain an equilibrium between using such principles and being too influenced by such principles in the modeling process. Thus, the bio-ontology development should follow a minimal commitment policy.

In the case of OBO bio-ontologies, there is a set of relationships, collected in the Relation Ontology (RO; Smith *et al*., 2005), that can be used in our bio-ontology. The use of such relations favors the integration with other bio-ontologies that also use RO, as, for example, the `participates_in` relation in our bio-ontology will be the same `participates_in` relation present in such other bio-ontologies. Therefore, bio-ontologies using such relations can be efficiently integrated and queried. Also, the RO relations have a precise semantic definition, saving time for the bio-ontologist, as there is no need to define the relations of the bio-ontology (if satisfied with the RO definition).

The use of an Upper Level Ontology (ULO), deeply related with the use of a set of relationships like RO, is also a recommended ontological practice. A ULO is generally an ontology with a few concepts that sits on the upper levels of the bio-ontology we are building, providing basic distinctions of types of concepts, like process vs. thing, self standing vs. refining entity, and so on. A ULO not only helps in integration with other bio-ontologies that are based in the same ULO, but also helps in building a sound and modular bio-ontology by creating

a cleaner structure with explicit distinctions. One of the most used ULOs in bio-ontologies is the Basic Formal Ontology (BFO; Grenon *et al*., 2004).

3.3.4 Knowledge acquisition

There are different ways of populating our bio-ontology with knowledge, described as follows. These methods are not disjoint; they can be used in a complementary manner.

An ideal method for obtaining the knowledge is to elicit it directly from the domain experts or the prospective users of our bio-ontology. Knowledge can also be obtained from extant resources. For example, data can be integrated from different resources in our bio-ontology, or knowledge from other bio-ontologies can be reused. Reusing content of other bio-ontologies is important to ease development and create a useful bio-ontology, since such a bio-ontology will be more interoperable with other resources. The OBO foundry ontologies offer a wealth of content that can be reused and extended with new axioms and entities. For example, that is the strategy followed in the creation of CCO (Antezana *et al*., 2009b).

3.3.5 Ontology design patterns (ODPs)

ODPs are solutions for common modeling problems that appear when building ontologies (Egaña Aranguren *et al*., 2008b). Thus, an ODP solves a concrete problem efficiently, as the ODP has been tested by a community of ontologists, and agreed to be an efficient modeling solution. Each ODP is thoroughly documented, clearly stating the requirements that the use of the ODP fulfills; that is, the problem that it solves. An ODP is like a 'cooking recipe' of how to create axioms that perform a given function within an ontology. Therefore, a bio-ontologist need only explore ODPs and apply the appropriate one in the bio-ontology being built. For example, in the case of the Value Partition ODP (Figures 3.8 and 3.9), such an ODP solves the problem of how to represent a feature that has only certain values (e.g., the height of a person can only be tall, medium or short). Ideally, if a bio-ontologist is confronted with the problem of representing such structure in a bio-ontology, he or she will explore ODP catalogs (see below), read the documentation, and, as the Value Partition ODP fulfills his or her requirements, apply it in the bio-ontology. Following such a procedure the bio-ontologist saves a lot of time, as many axioms are applied automatically in the bio-ontology.

ODPs are presented as fragments of ontologies that solve a concrete modeling problem, as a concrete set of axioms, but with an abstract structure: when applied in the ontology, such axioms relate the actual entities of the ontology. Therefore, ODPs can also be regarded as modules of ontologies to be applied 'off the shelf': an ontology can rapidly be built by applying a collection of ODPs.

Using ODPs in the development of an ontology makes such development faster, more consistent, and explicit. The resulting bio-ontologies have a richer

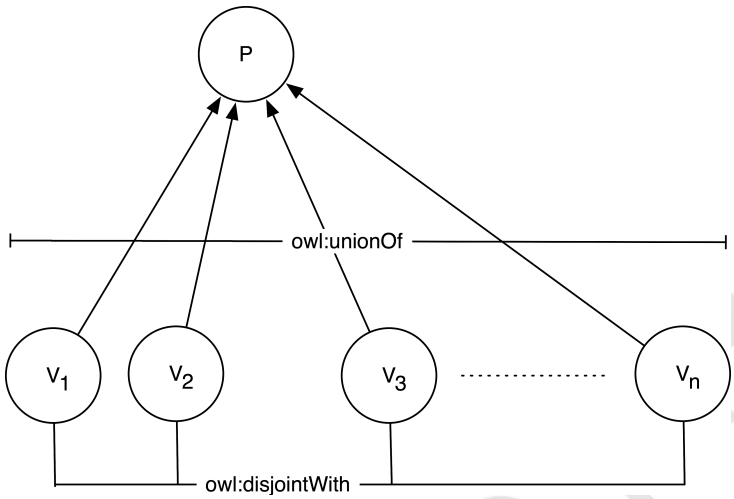

Figure 3.8 Abstract representation of the Value Partition ODP. This ODP solves a concrete problem; namely, how to represent exhaustive sets of values in OWL. `P` *can be any feature (regulation, color, height, etc.), and* `V` *any value (positive or negative; red, blue or white; etc.). This abstract structure is presented with documentation that explains how the ODP can be used (e.g., motivation, structure, elements, implementation, and result).*

axiomization, obtained with less effort, enhancing automated reasoning. They are also more reusable and interoperable with other bio-ontologies.

There are two main catalogs where ODPs can be obtained (http://odps.sf.net/, http://ontologydesignpatterns.org). Once an ODP has been chosen, there are different methods for applying it. The ODP can be directly imported into the ontology, manually recreated, or applied with ODP-oriented tools like the NeOn toolkit (http://neon-toolkit.org) or the Ontology PreProcessor Language (OPPL; http://oppl.sourceforge.net/).

3.3.6 Ontology evaluation

Ontology evaluation is a controversial issue, and there is a wealth of methodologies to choose from, depending on the needs of the project. Three main and complementary categories can be identified, according to the aims of the evaluation process: ranking, correctness, and quality.

Ranking approaches pursue the selection of the best ontology for a particular task, so they apply criteria that focus on that particular task. Ranking strategies may be driven by users, experts, and so on. Bio-ontologies can get different results using different ranking strategies, as different quality aspects are measured. For example, in Aktiverank (Alani *et al*., 2006), ontologies are ranked against search terms, so that the best ontology is the one that best matches the query. For this

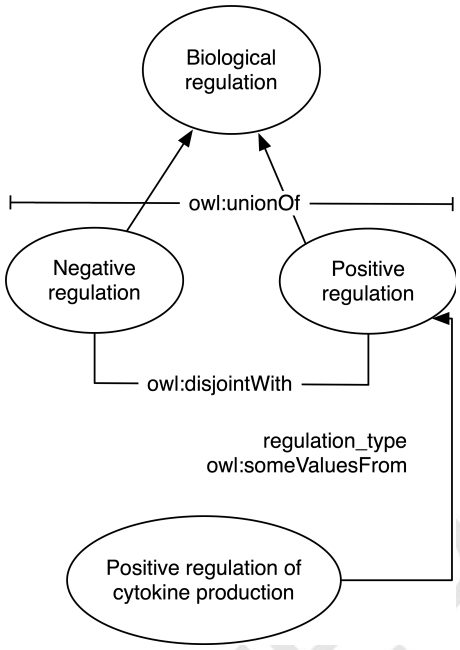

Figure 3.9 Application of the Value Partition ODP in GO. Thus, the abstract structure from Figure 3.8 is converted to a concrete structure with concrete entities, and linked to the rest of the bio-ontology by the `regulation_type` *relation (with the existential qualifier).*

purpose, quantitative metrics such as the coverage of an ontology for the given search term, the number of connections (relations, subclasses, superclasses, and siblings), or the closeness of the classes that matches the search terms in the ontology are used.

Correctness approaches determine the quality of a bio-ontology by applying formal theories. The most relevant approach is provided by Ontoclean (Guarino and Welty, 2004), which checks for the formal correctness of the taxonomy, based on rigidity, identity, unity, and dependence principles.

Quality approaches provide frameworks that are based on a series of qualitative and quantitative criteria that can be organized in quality dimensions. The goal of such approaches is to provide an overview of the strengths and weaknesses of the bio-ontologies in the particular quality dimensions rather than finding the best one for a particular task. Quality approaches are likely to include criteria that cannot always be optimized simultaneously, and this makes their application more complex. In (Fernández-Breis *et al*., 2009), an ISO 9126-based framework was proposed, comprised of seven quality dimensions: structural, functionality, reliability, usability, efficiency, maintainability, and quality in use.

3.3.7 Documentation

Most KR languages allow the inclusion of information to axioms and entities in the form of annotations that are not processed by the automated reasoner[6]. For example, OWL allows one to create custom annotation properties or use the already defined `rdfs:comment`, `rdfs:label`, or the Dublin Core (http://dublincore.org/) annotation properties. The OBO format has its own set of annotations tailored to the OBO community needs. Annotations are usually used to capture information that cannot be represented in axioms, to capture information that should not be represented in axioms (e.g., the name of an entity in different languages) or to express facts about the modeling in natural language (e.g., the rationale for modeling decisions).

It is important to capture as much information as possible in annotations, as it will be used by other developers or users. Such annotations should also be as structured as possible: for example, the GO term names are syntactically very repetitive (Ogren *et al*., 2004), which helps in computationally processing them (Egaña Aranguren *et al*., 2008a).

3.4 Conclusion

The Life Sciences Semantic Web (LSSW) faces many challenges. KR languages with precise semantics like OWL, being powerful and robust solutions for a truly distributed and automatic knowledge management, are not free of problems. The increasing volume of available data and supporting bio-ontologies reveals limitations in terms of performance, especially regarding automated reasoning and the management of KBs. Performance issues are expected to be solved as the technology evolves. However, there are also problems in the 'social' side of bio-ontology creation, the main one being the lack of agreement in modeling principles: for example, there is not even a consensus on how to represent a concept as important and basic as the one of species (Schulz *et al*., 2008). Such lack of agreement is a community problem, but there are practices, like the use of ODPs, that can contribute to its solution.

Even taking into account these problems, the LSSW offers an increasing number of examples that make good on its promise to help in the information management of biological knowledge, and to support advanced queries that demonstrate the power of semantic data integration.

The adoption of a precise semantics opens new paradigms of biological research, like the Semantic Systems Biology (SSB) approach (Antezana *et al*.,

[6] The term 'annotation' has a somewhat different meaning in bioinformatics and KR. In bioinformatics, an annotation is information attached to biological data, such as the molecular function of a gene product. In KR, an annotation is extralogical information added to an axiom or an entity of an ontology, usually using natural language. We are using the KR meaning throughout the chapter.

2009c). SSB is a systems biology approach that combines Semantic Web technologies for analyzing data and formalized knowledge to engineer biological system models. Kitano's Systems Biology paradigm (Kitano, 2002) hinges on mathematical model-based system behavior predictions, or hypotheses, and validation in new experiments. In SSB, data and new knowledge are (automatically) checked for consistency against existing knowledge, and queries and automated reasoning on semantically integrated knowledge are used to extract new knowledge and hypotheses.

Post *et al*. applied such an approach to study the role of histone modification in gene expression regulation (Post *et al*., 2007). In that use case as well as in other efforts such as the YeastHub (Cheung *et al*., 2005), CViT (Deisboeck *et al*., 2007), and the Cell Cycle Ontology (Antezana *et al*., 2009b), the workflow of an SSB approach was followed. Some other initiatives are NeuroCommons (http://neurocommons.org), focused on neuroscience, and the SSB portal (www.semantic-systems-biology.org). All these initiatives demonstrate the added value that the SSB approach can offer to the understanding of biological systems.

This chapter has provided a brief overview of the extant technologies and tools to build bio-ontologies, as well as real bio-ontology examples and pointers to the future of the LSSW, like SSB. Also, it has highlighted the most important issues and practices that should be taken into account in order to create a useful bio-ontology with the least possible distress. Creating proper bio-ontologies is a very hard task; however, it is even harder to manage biological data, information, and knowledge efficiently without them.

3.5 Acknowledgements

Mikel Egaña Aranguren was funded by the Autonomous Community of the Region of Murcia, Spain (BIO-TEC 06/01-0005). Vladimir Mironov was funded by FUGE Mid-Norway.

3.6 References

Alani, H., Brewster, C., and Shadbolt, N. (2006) Ranking ontologies with AKTiveRank. International Semantic Web Conference (ISWC 2006), Athens, GA, USA, 5–9 November 2006.

Antezana, E., Egaña, M., De Baets, B., *et al*. (2008) ONTO-PERL: an API supporting the development and analysis of bio-ontologies. *Bioinformatics*, **24**(6), 885–7.

Antezana, E., Kuiper, M., and Mironov, V. (2009a) Biological knowledge management: the emerging role of the Semantic Web technologies. *Brief. Bioinformatics*, **10**(4), 392–407.

Antezana, E., Egaña, M., De Baets, B., *et al*. (2009b) The Cell Cycle Ontology: an application ontology for the representation and integrated analysis of the cell cycle process. *Genome Biol.*, **10**, R58.

Antezana, E., Blondé, W., Egaña, M., *et al*. (2009c) BioGateway: a Semantic Systems Biology tool for the life sciences. *BMC Bioinformatics*, **10**(Suppl 10), S11.

Baader, F., Calvanese, D., McGuinness, D., *et al*. (eds) (2003) *The Description Logic Handbook*. Cambridge University Press, Cambridge.

Bard, J.B.L, Rhee, S.Y., and Ashburner, M. (2005). An ontology for cell types. *Genome Biol.*, **6**, R21.

Bodenreider, O. and Stevens, R. (2006) Bio-ontologies: current trends and future directions. *Brief. Bioinformatics*, **7**(3), 256–74.

Cheung, K.H., Yip K.Y., Smith, A., *et al*. (2005) YeastHub: a semantic web use case for integrating data in the life sciences domain. *Bioinformatics*, **21**(Suppl 1), i85–96.

Deisboeck, T.S., Zhang, L., and Martin, S. (2007) Advancing cancer systems biology: introducing the Center for the Development of a Virtual Tumor, CViT. *Cancer Inform*, **5**, 1–8.

Egaña Aranguren, M., Wroe, C., Goble, C., and Stevens, R. (2008a) In situ migration of handcrafted ontologies to Reason-able Forms. *Data Knowl. Eng.*, **66**(1), 147–62.

Egaña Aranguren, M., Antezana, E., Kuiper, M., and Stevens, R. (2008b) Ontology Design Patterns for bio-ontologies: a case study on the Cell Cycle Ontology. *BMC Bioinformatics*, **9**(Suppl 5), S1.

Fernández-Breis, J.T., Egaña Aranguren, M., and Stevens, R. (2009) A quality evaluation framework for bio-ontologies. International Conference on Biomedical Ontology (ICBO 2009), Buffalo, USA, 24–26 July 2009.

Gene Ontology Consortium (2000) Gene Ontology: tool for the unification of biology. *Nat. Genet.* **25**, 25–9.

Golbreich, G., Horridge, M., Horrocks, I., *et al*. (2007) OBO and OWL: Leveraging Semantic Web technologies for the life sciences. International Semantic Web Conference (ISWC 2007), Busan, Korea, 11–15 November 2007.

Good, B.M. and Wilkinson, M.D. (2006) The life sciences Semantic Web is full of creeps! *Brief. Bioinformatics*, **7**(3), 275–86.

Grenon, P., Smith, B., and Goldberg, L. (2004) Biodynamic Ontology: applying BFO in the biomedical domain, in *Ontologies in Medicine*, (ed. D.M. Pisanelli), IOS Press, pp. 20–38.

Guarino, N. and Welty, C.A. (2004) An overview of OntoClean, in *Handbook on Ontologies* (eds S. Staab and R. Studer), Springer, pp. 151–72.

Horridge, M., Drummond, N., Goodwin, J., *et al*. (2006) The Manchester OWL syntax. OWL: Experiences and Directions (OWLED 06), Athens, GA, USA, 10–11 November 2006.

Kitano, H. (2002) Systems biology: a brief overview. *Science*, **295**(5560), 1662–4.

Miñarro-Gimenez, J.A., Madrid, M., and Fernández-Breis, J.T. (2009) OGO: an ontological approach for integrating knowledge about orthology. *BMC Bioinformatics*, **10**(Suppl 10), S13.

Mungall, C.J. (2004) OBOL: integrating language and meaning in bio-ontologies. *Comp. Funct. Genomics*, **5**(6-7), 509–20.

Mungall, C.J., Gkoutos, G.V., Smith, C.L. *et al*. (2010) Integrating phenotype ontologies across multiple species. *Genome Biol.*, **11**, R2.

Ogren, P.V., Cohen, K.B., Acquaah-Mensah, G.K., *et al*. (2004) The compositional structure of Gene Ontology terms. Pacific Symposium on Biocomputing (PSB 04), Big Island, Hawaii, USA, 6–10 January 2004.

Post, L.J.G., Roos, M., Marshall, M.S., *et al*. (2007) A semantic web approach applied to integrative bioinformatics experimentation: a biological use case with genomics data. *Bioinformatics*, **23**(22), 3080–7.

Schulz, S., Stenzhorn, H., and Boeker, M. (2008) The ontology of biological taxa. *Bioinformatics*, **24**(13), i313–21.

Smith, B., Ceusters, W., Klagges, B., *et al*. (2005) Relations in Biomedical Ontologies. *Genome Biol.*, **6**, R46.

Smith, B., Ashburner, M., Rosse, C., *et al*. (2007) The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.*, **25**, 1251–5.

Stevens, R. and Lord, P. (2008) Application of ontologies in bioinformatics, in *Handbook on Ontologies in Information Systems*, 2nd edn (eds S. Staab and R. Studer), Springer, pp. 735–56.

Stevens, R., Egaña Aranguren, M., Wolstencroft, K., *et al*. (2007) Using OWL to model biological knowledge. *Int J Hum Comput Stud.*, **65**(7), 583–94.

Queries in Chapter 3

Q1. We have shortened the running head as it exceeds the width of the page. Please check