Owlifier: Creating OWL-DL Ontologies from Simple Spreadsheet-Based Knowledge Descriptions☆

Shawn Bowers[a,*], Joshua S. Madin[b], Mark P. Schildhauer[c]

[a] *UC Davis Genome Center*
[b] *Dept. of Biological Sciences, Macquarie University, Australia*
[c] *National Center for Ecological Analysis and Synthesis, UC Santa Barbara*

Abstract

Discovery and integration of data is important in many ecological studies, especially those that concern broad-scale ecological questions. Discovery and integration is often a difficult and time-consuming task for researchers, in part because informal, ambiguous, and sometimes inconsistent terms are used to describe data semantics. Ontologies can help address this problem by providing a means to define ecological concepts to more consistently annotate, relate, and search for data sets. However, unlike in molecular biology or biomedicine, few ontology development efforts exist within ecology. Ontology development often requires considerable expertise in ontology languages and development tools, which is often a barrier for ontology creation in ecology. In this paper we address this problem by providing an approach for ontology creation that allows ecologists to use common spreadsheet tools to describe different aspects of an ontology. We present conventions for creating, relating, and constraining concepts through spreadsheets, and provide software tools for converting these ontologies into equivalent OWL-DL representations. We also consider inverse translations, i.e., to convert ontologies represented using OWL-DL into our spreadsheet format. Our approach allows large lists of terms to be easily related and organized into concept hierarchies, and generally provides a more intuitive and natural interface for ontology development by ecologists.

1. Introduction

Within the fields of molecular biology and biomedicine considerable effort has gone into developing ontologies for improving data discovery and integration [2, 1]. While similar benefits can be obtained for ecological data, far fewer efforts exist to develop richer and more consistent terminologies of ecology concepts [8, 10]. The use of formal ontologies can significantly enhance metadata descriptions of ecological data. Annotating data with ontology terms can both

[*] Corresponding author

help users interpret data as well as enable advanced capabilities for data discovery and integration, e.g., by exploiting subsumption and part-of hierarchies as well as more formal constraints such as cardinality restrictions on properties and term equivalence.

Efforts to engage scientists in the development of ontologies typically leverage the W3C Web Ontology Language (OWL) [15] as a standard XML syntax for representing and sharing ontologies. A key advantage of OWL is that it is supported by a wide range of generic tools, including editors [7, 6], reasoning systems [14, 16], query languages [11, 9], and storage technologies [4, 3]. Most of these tools, however, are primarily targeted at experts in knowledge engineering and software development familiar with the underlying description logic semantics of OWL-DL [5]. This is especially true with ontology editors (such as Protege, SWOOP, etc.), which allow for very detailed ontology specifications, but at the same time require a considerable amount of understanding of the underlying ontology formalisms and syntax. Thus, we see the lack of suitable ontology editing tools for scientists as one of the major barriers for more wide-scale adoption of ontologies in ecology.

This paper presents a novel approach for ontology creation that aims at being more intuitive for ecologists and that can be used to rapidly construct large ontologies for describing scientific data. Our approach is to allow scientists to use common spreadsheet-based tools to describe, in an intuitive way, different aspects of an ontology, and then to take these descriptions and convert them into full-fledged OWL ontologies using a software application called owlifier. An owlifier spreadsheet consists of a set of *blocks* that have a predefined template structure for users to fill in. Each non-empty row in an owlifier table constitutes a block. Each block defines different aspects of an ontology including ontology classes, subclasses, synonyms, and properties. We also provide blocks for plain-text descriptions of classes and properties, and for referencing one or more existing ontologies (e.g., to extend an existing ontology or to define ontology articulations). Blocks can be sparse (inheriting from previous blocks), which can further simplify the creation of large ontologies.

While not as expressive as OWL-DL, our approach can produce ontology structures essential for enhanced data discovery and integration, while at the same time provide a more accessible user interface for ecologists. Further, our approach can be used to rapidly construct class hierarchies from long lists of keywords using familiar spreadsheet software. For instance, an ecologist can easily list (or import) a set of terms, and then incrementally organize these into a class hierarchy, define properties and constraints, etc. We have used the owlifier approach with ecologists and evolutionary biologists working with trait data, and found that it enabled them to quickly and easily comprehend and construct ontologies.

The rest of this paper is organized as follows. In Sect. 2 we describe the basic syntax and semantics of owlifier. We define blocks that support a large subset of OWL-DL and that also generally follow the ontology creation guidelines defined in [13]. We also simplify certain aspects of ontology creation using OWL-DL, e.g., by assuming classes are disjoint by default (unless specified otherwise) and by applying implicit property restriction closures [13]. In Sect. 3 we describe additional characteristics of owlifier and discuss issues with respect to classification and reasoning. In Sect. 4 we briefly describe the owlifier implementation, and conlcude in Sect. 5 with related and future work. In general, the goal of owlifier is not to support all constructs in OWL-DL, but instead to provide a higher-level ontology syntax (via spreadsheet blocks) that is easy for ecologists to use and understand while also providing the necessary constructs for developing typical ecological ontologies. By compiling owlifier to OWL-DL, we also allow for experts to refine and extend the ontology using more advanced ontology editing tools if necessary.

2. Owlifier Syntax and Semantics

An owlifier table defines an OWL-DL [15] ontology through a set of *blocks* representing one or more ontology definitions. Each non-empty row in an owlifier table corresponds to a block. The type of the block is given in the first column of the row. We describe each type of block supported by owlifier below. Here we assume that if any of the properties or concepts used in a block are defined, i.e., are not imported from another ontology, then they are added to the ontology being defined by the owlifier table.

Import Block. Import blocks assign namespace labels to external ontologies. Each external ontology is imported into the current ontology. We refer to the ontologies of import blocks as *imported ontologies.* Using import blocks, classes and properties of imported ontologies can be used within other blocks of an owlifier table. Import blocks take the form

`import`	N	U

where N is a namespace label and U is an OWL ontology URI. Classes and properties from imported ontologies are referenced by prefixing the namespace label N to the corresponding class or property name in the normal way. The following block imports the "earth realm" ontology from the set of SWEET ontologies [12]

```
import sweet http://sweet.jpl.nasa.gov/ontology/earthrealm.owl
```

where the ontology class denoting marine ecosystems can be refered to from within the corresponding owlifier table using the syntax, e.g., `sweet:MarineEcosystem`. Because this class refers to a class in another ontology, we call it an "imported" class.

Entity Block. Entity blocks introduce new OWL classes and specify subclass relationships. Imported classes may also be used within a concept block by prefixing the class name with a namespace label, e.g., when an ontology is being extended. We use the term 'entity to avoid confusion with the use of the term 'class' in biological taxonomies and because 'entity' is used in [**?**] to define ecological concepts. Entity blocks take the form

$$\begin{array}{|c|c|c|c|c|}\hline \texttt{entity} & C_1 & C_2 & \dots & C_n \\ \hline\end{array} \qquad (n \geq 1)$$

where each class C_i is asserted in the current ontology to subsume a class C_{i+1}, for $1 \leq i < n$. Each C_i in a concept block induces the DL axiom

$$C_i \sqsubseteq C_{i+1}.$$

If both C_i and C_{i+1} are imported classes, we say that the block defines an "articulation" (i.e., mapping) between the two classes. The following concept block defines a simple subclass hierarchy

```
entity PhysicalFeature AquaticPhysicalFeature River
```

which states that river is a subclass of aquatic physical feature and that aquatic physical feature is a subclass of physical feature.

Synonym Block. Synonym blocks define an equivalence relationship between classes. A synonym block has the form

$$\begin{array}{|c|c|c|c|c|}\hline \texttt{synonym} & C_1 & C_2 & \dots & C_n \\ \hline\end{array} \qquad (n \geq 2)$$

where each class C_i is asserted as being equivalent to class C_{i+1} in the current ontology, for $1 \leq i < n$. For example, the block

```
synonym Maize Corn
```

defines maize and corn as synonym (i.e., equivalent) classes.

Overlap Block

Except in certain situations (described further below), classes are assumed to be disjoint. Overlap blocks explicitly relax this assumption for a given set of classes by stating that a set of classes may have overlapping instances. An overlap block has the form

$$\begin{array}{|c|c|c|c|c|}\hline \texttt{overlap} & C_1 & C_2 & \dots & C_n \\ \hline\end{array} \qquad (n \geq 2)$$

where each class C_i is allowed to share instances with each concept C_j, for $1 \leq i, j \leq n$. In particular, C_i and C_j are not defined to be disjoint classes in the current ontology.

— an example here —

Relationship Block

Relationship blocks define required *object* properties of classes. A property block has the form

$$\begin{array}{|c|c|c|c|c|c|}\hline \texttt{relationship} & R & C_1 & C_2 & \dots & C_n \\ \hline\end{array} \qquad (n \geq 2)$$

where R is an object property and each C_i is a class, for $1 \leq i \leq n$. For every class C_i, the property block induces the DL axiom

$$C_i \sqsubseteq \exists R.C_{i+1}$$

stating that each instance of C_i has, amongst possibly other things, a relationship through R to some instance of C_{i+1}. The following example

need example here

Transitive Block

Transitive blocks state that a property is transitive. That is, if P is transitive and a concept instance O_1 is related to an instance O_2 by P, and O_2 is related to an instance O_3 by P, then O_1 is also by definition related to O_3 by P. A transitive block has the form

`transitive`	R	C_1	C_2	$\ldots$	C_n

$(n \geq 2)$

where P is an object property. For example, the block

```
transitive hasPart Body Head Eye Retina
```

states that a body has at least one head, a head has at least one eye, and an eye has at least one retina.

Attribute Block.. Attribute blocks are used to define the required *datatype* properties of concepts. An attribute block has the form

`attribute`	P	D	C_1	C_2	$\ldots$	C_n

$(n \geq 1)$

where P is a datatype property, each C_i is a concept for $1 \leq i \leq n$, and D is a datatype (`anyValueType`, `string`, `int`, etc.). For every concept C_i, the property block induces the DL axiom

$$C_i \sqsubseteq (\exists P.D)$$

stating that each instance of C_i has, amongst possibly other things, a relationship through P to a data value of type D.

Value Block.. Value blocks define required datatype property *constant values* for concepts. A value block has the form

`value`	P	V	C_1	C_2	$\ldots$	C_n

$(n \geq 1)$

where P is a datatype property, C_i is a concept for $1 \leq i \leq n$, and V is a datatype value. For each concept C_i, the value block induces the DL axiom

$$C_i \sqsubseteq (V \in P)$$

stating that each instance of C_i has a value V for property P. The value restrictions stated by value blocks are often used for defining so-called *value partitions* [**?**].

Inverse Block.. Inverse blocks state that two object properties are inverses of each other. That is, for inverse properties P_1 and P_2 and concept instances O_1 and O_2, if $P_1(O_1) = O_2$, then $P_2(O_2) = O_1$. An inverse block has the form

`inverse`	P_1	P_2

where P_1 and P_2 are object properties.

Minimum Block.. Minimum blocks state the minimum number of properties P an instance of a concept may have. Minimum blocks have the form

`minimum`	P	N	C_1	C_2	$\dots$	C_m

where N is the minimum number of properties P that instances of concept C_1 may have to instances of concept C_2, C_2 to C_3, and so on. A cardinality block induces the DL axiom

$$C_i \sqsubseteq (\leq NP.C_{i+1})$$

stating that each instance of C_i must be related to at least N unique instances of C_{i+1} via P. For example, the blocks

```
minimum hasPart 1 Body Head
minimum hasPart 2 Head Eye
```

states that a body has at least one head and at least two eyes.

Maximum Block.. Maximum blocks state the maximum number of properties P an instance of a concept may have. Maximum blocks have the form

`maximum`	P	N	C_1	C_2	$\dots$	C_m

where N is the maximum number of properties P that instances of concept C_1 may have to instances of concept C_2, C_2 to C_3, and so on. A cardinality block induces the DL axiom

$$C_i \sqsubseteq (\geq NP.C_{i+1})$$

stating that each instance of C_i may be related to at most N unique instances of C_{i+1} via P. For example, the blocks

```
maximum hasPart 1 Body Head
maximum hasPart 2 Head Eye
```

states that a body has at least one head and at least two eyes.

Sufficient Block.. Sufficient blocks state that any instance having a property P to an instance of a concept C_2 is a sufficient condition for being an instance of a concept C_1. A sufficient block has the form

`sufficient`	C_1	P	C_2

where C_1 is the target concept (i.e., denoting the concept definition), P is the sufficient property, and C_2 is the sufficient concept. A sufficient block induces the DL axiom

$$C_1 \equiv \exists P.C_2$$

Sufficient blocks provide a mechansism to construct simple class definitions (i.e., classes defined precisely by other classes), primarily for use with value partitions. [NOTE: these should be anded together?]

Description Block.. Description blocks assign plain-text definitions to concepts and properties. A description block has the form

`description`	T	S

where T is either a property or a concept and S is a description string.

Note Block.. Note blocks add comments to the current ontology, and are ignored by owlifier. A note block has the form

`note`	S

where S is a comment string.

*** Say something about relaxing block syntax ... to make it easier to specify ontologies. Also, allow blocks to be given in any order.

3. owlifier Properties and Reasoning

Some desirable properties:

- non-ambiguous (no block leads to ambiguous DL axioms)
- "reasonable" (not everything has to be explicitly stated)
- ???

4. owlifier Implementation

Flags:

- delimeter characters
- perform classification
- warnings
- owlifier to owl and owl to owlifier
- ???

5. Conclusion

References

[1] M. Ashburner, *et al.* Gene ontology: tool for the unification of biology. *Nat. Genet.*, 25:25–29, 2000.

[2] J. Bard and S. Rhee. Ontologies in biology: design, applications, and future challenges. *Nat. Rev. Genet.*, 5:213–221, 2004.

[3] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. In *International Semantic Web Conference (ISWC)*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[4] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *International Conference on the World Wide Web (WWW)*, pages 74–83. ACM, 2004.

[5] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2008. in press.

[6] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, and J. Hendler. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*, 4(2), 2005.

[7] H. Knublauch, M. A. Musen, and A. L. Rector. Editing description logic ontologies with the protégé owl plugin. In *International Workshop on Description Logics (DL)*, volume 104 of *CEUR Workshop Proceedings*, 2004.

[8] J. Madin, S. Bowers, M. Schildhauer, and M. Jones. Advancing ecological research with ontologies. *Trends in Eco. and Evol.*, 23(3):159–168, 2008.

[9] B. Motik, U. Sattler, and R. Studer. Query answering for owl-dl with rules. *J. Web Sem.*, 3(1):41–60, 2005.

[10] C. Parr and M. Cummings. Data sharing in ecology and evolution. *Trends Ecol. Evol.*, 20:362–363, 2004.

[11] E. Prudhommeaux and A. Seaborne, editors. *SPARQL Query Language for RDF.* W3C. World Wide Web Consortium (W3C), January 2008.

[12] R. Raskin. Semantic web for earth and environmental terminology (sweet). http://sweet.jpl.nasa.gov/.

[13] A. L. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In *EKAW*, pages 63–81, 2004.

[14] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *J. Web Sem.*, 5(2):51–53, 2007.

[15] M. K. Smith, C. Welty, and D. L. McGuinness, editors. *OWL Web Ontology Language Guide.* W3C. World Wide Web Consortium (W3C), February 2004.

[16] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297, 2006.