A Formal Description of Ontology Change in OWL

John Avery, John Yearwood

University of Ballarat, Australia

E-mail: j.avery@ballarat.edu.au, j.yearwood@ballarat.edu.au

Abstract

There are three main activities involved in managing ontology change. Firstly we need to identify changes, secondly describe these identified changes, and finally describe and handle the ramifications of the changes. In previous work we have presented a language (DOWL) for describing ontology change and in this paper we demonstrate how changes described in this language can be represented in the RDF abstract syntax which enables us to describe the ramifications of a change in a formal manner. This formalism can provide the basis for an automated ontology change management system.

1. Introduction

In order for the Semantic Web vision to be realised it must provide a robust platform for the interchange of semantically rich information. In order to achieve this robustness changes to the ontologies that provide the semantics of the Semantic Web must be effectively managed. This task is far from trivial as the ontologies on the semantic web will be complex and richly interrelated.

Ontologies will change in reflection of real world knowledge changes, and in reflection of the inaccuracies arising from the inherent difficulty of the knowledge acquisition process. Although the recognition and description of these changes is an important problem discussed in [4, 6], in this paper we focus on the ramifications of these changes once they have been recognised and described. We formally describe the actions that need to be taken in order to make all things dependent upon the pre-change ontology conform to the definitions in the changed ontology. A similar treatment has been given to change management in database schema [3], but the problem of ontology change management is a significantly different and richer problem [5]. By describing these changes formally we pave the way for the automation of these changes in an ontology change management system, with well defined outcomes in terms of validity, consistency, and the like.

Our discussion of ontologies is set in the context of the Semantic Web so we describe ontologies using the Web Ontology Language (OWL), this language is overviewed in section 2. In order to describe the ramifications of change we need a language for describing ontology change, we do so using Dynamic OWL (DOWL) a language we developed for this purpose which is described in section 3. The semantics presented in this paper can be considered an extension of DOWL.

Section 4 describes in more detail what we mean by the ramifications of an ontology change and in Section 5 we introduce the DOWL semantics and then go on to use them to describe the ramifications of several, varied, ontology changes. We conclude with a summary of the paper and indicate the direction of future work.

2. OWL

This section provides an overview of OWL Abstract Syntax and Semantics for a description of the language see [8]. The fundamental building blocks of an OWL ontology are classes, properties and individuals. Although OWL comes in various flavours (OWL Full, OWL DL, and OWL Lite) we discuss only the unrestricted flavour (OWL Full), the most notable aspect of OWL Full is that there is no separation between classes, properties and individuals. An object may, for example, be both a class and a property.

2.1. OWL Semantics and Abstract Syntax

There is a high level abstract syntax for OWL. A model-theoretic semantics is also given to provide a formal meaning for OWL ontologies as RDF graphs (for OWL Full). In this section we give the definitions from the RDF-Compatible Model-Theoretic Semantics for OWL which will be needed for the later discussion. We intend only to give the reader some familiarity with the model-theoretic semantics expressed in this form, it is not a definitive guide. For a full treatment see [9].

2.2. Defintions (from [9])

From the RDF semantics [7] for V a set of URI references and literals containing the RDF and RDFS vocabulary and D a datatype map, which is a partial mapping from URI references to datatypes. A D -interpretation of V is a tuple $I = \langle R_I, P_I, EXT_I, S_I, L_I, LV_I \rangle$

- R_I is the domain of discourse or universe, i.e., a nonempty set that contains the denotations of URI references and literals in V .
- P_I is a subset of R_I , the properties of I .
- EXT_I is used to give meaning to properties, and is a mapping from P_I to $P(R_I \times R_I)$.
- S_I is a mapping from URI references in V to their denotations in R_I .
- L_I is a mapping from typed literals in V to their denotations in R_I .
- LV_I is a subset of R_I that contains at least the set of Unicode strings, the set of pairs of Unicode strings and language tags, and the value spaces for each datatype in D .
- The set of classes C_I is defined as $C_I = \{x \in R_I \mid \langle x, S_I(\text{rdfs:Class}) \rangle \in EXT_I(S_I(\text{rdf:type}))\}$, and the mapping $CEXT_I$ from C_I to $P(R_I)$ is defined as $CEXT_I(c) = \{x \in R_I \mid \langle x, c \rangle \in EXT_I(S_I(\text{rdf:type}))\}$.
- D -interpretations must meet several other conditions as detailed in [7].

3. Modelling Change

In [1] we presented a generic ontology model which we use to model change in an ontology. Figure 1 depicts our generic model. The key element of the model is the separation of the domain being modeled and the ontology through a layer we call the model. This separation facilitates the recognition of two distinct facets of ontology change; there can be changes in *how* an entity is modelled (a *transformation*), or changes in *which* entities are modelled (a *translocation*).

Transformations occur when an entity has been modelled incorrectly or inadequately, for example we may transform an entity representing a man to include the notion that the class of men is disjoint from that of women. Transformation necessarily implies a change in the ontological construct describing the representation being changed.

Translocation represents changes in the intent of the ontologies designer. For example a web services location may

Figure 1. The Generic Ontology Model.

be better represented by a URI than by a URL. Translocations are not described in an ontological construct (this does not mean they do not have ramifications).

3.1. DOWL

In [2] we go on to list the various ways in which an ontology can change and describe these changes using the notions of transformation and translocation. A key realisation is that many types of ontology change are dependant on the ontology language being used. As our work is set in the context of the Semantic Web we go on to elaborate an atomic set of operators for describing change in OWL ontologies which we argue provides a necessary and sufficient set of operators for expressing changes in an OWL ontology. We express these operators in our DOWL language which is described in full at <http://uob-community.ballarat.edu.au/~javery/dowl/>. Table 1 lists a few of the DOWL operators with a brief description.

Operator	Description
dowl:add	Add an ontological construct to the ontology.
dowl:rename	Change the identifier of an ontological construct.
dowl:broaden	Broaden the notion of a representation.
dowl:isNotSameAs	Remove the equivalency between two constructs.
dowl:undefineAsObjectProperty	Make an ontological construct not an object property.

Table 1. A selection of DOWL operators.

4. Ramifications and the Semantics of Change

The most important reason for describing ontology change, is to be able to manage that change. A change in one ontology on the web may have repercussions on other parts of the semantic web, like ontologies that are dependant on the changed ontology, or documents that are annotated

using the changed ontology. We call these repercussions the ramifications of an ontology change. One of our goals is to formally describe the ramifications of each of the DOWL operators so that these ramifications can be handled automatically.

In essence everything that can be affected by an ontology change must be an ontology. Although we talk about documents that are annotated using an ontology, this annotation is in effect an ontology which is the only part of the document that has ontological significance. We consider documents to be a special case of ontologies which only allow certain types of ontological constructs: that is they can only include statements that affect our knowledge about individuals, not about classes and properties.

Proofs will be used in the semantic web to validate or support a claim, they will form an important part of the Semantic Web. Although as yet the Semantic Web has no language for expressing proof, it is reasonable to suggest that this language will consist of a chain of reasoning. Thus proofs will be ontologies with some additional semantics. However in order to be Semantic Web compliant it is not necessary to construct proofs only check existing ones. Thus to handle the ramifications of an ontology change on proof we only need to determine if the proof is still valid.

The upshot of this is that if we describe the ramifications of an ontology change on a dependant ontology we have described all possible ramifications of ontology change, the notable difference for proofs is that we cannot reannotate proofs as we are not able to construct proofs, only verify existing ones.

5. DOWL Semantics

5.1. A DOWL Interpretation

In this section we introduce the definitions that are used to describe ontology change and its ramifications. All of the notation from the OWL RDF-Compatible Model-Theoretic Semantics is assumed to be defined, and is different only when indicated. After these definitions we describe several ontology changes using the model-theoretic semantics and associated abstract syntax. We also indicate the implications of the change on annotations and dependant ontologies.

Definitions

- We represent the D-Interpretation of V before a change as a tuple $I = \langle R_I, P_I, EXT_I, S_I, L_I, LV_I \rangle$.
- We represent the D-Interpretation of V after a change (described by one DOWL operator) as a tuple $J = \langle R_J, P_J, EXT_J, S_J, L_J, LV_J \rangle$.
- V_{DOWL} is the dowl vocabulary.

- T_I is the mapping of denotations in R_I to URI references, it is the inverse of S_I .
- We represent a DOWL evolution as a tuple $E = \langle O_E, S_E, RI_E, RJ_E, MI_E, MJ_E \rangle$.
 - O_E is the set of denotational dOWL operators.
 - S_E is a mapping from URI references in V_{DOWL} to their denotations in O_E .
 - RI_E is the set of denotational objects from the unevolved ontology, $RI_E \subseteq R_I$.
 - RJ_E is the set of denotational objects from the evolved ontology, $RJ_E \subseteq R_J$.
 - MI_E is a mapping between from O_E and RI_E .
 - MJ_E is a mapping between O_E and RJ_E .
- We represent the set of OWL statements from a dependant ontology as a tuple $D = \langle R_D, P_D, EXT_D, S_D, L_D, LV_D \rangle$.
- We represent the set of OWL statements from a proof ontology as a tuple $\rho = \langle R_\rho, P_\rho, EXT_\rho, S_\rho, L_\rho, LV_\rho \rangle$.
- In order to describe the annotational changes that need to be made we define a set of OWL statements (expressed as three URI's forming an rdf triple) that need to be added to the ontology (A_A), a set of URI references that need to be removed (A_R), and a mapping (A_C) which maps between the URI references before a change, to the URI references after a change. Proofs are not re-annotated, as this may cause them to become invalid, according to the as yet unspecified inference rules of OWL.
- Changes to an ontology may also make the ontology invalid, we list the conditions which in conjunction with an ontology change of a specific type will cause the ontology to be invalid.
- In order to describe translocational changes we define a mapping F_P which relates literals in I to literals in J for the property P . In this way any kind of translocation can be handled. F_P may be a function (between degrees celcius and degrees fahrenheit for example). F_P in many cases will map values to themselves.

5.2. Formal Descriptions and Ramifications of selected DOWL Operators

In this section we formally describe each of the DOWL operators presented in Table 1 and present their ramifications using the notation described above.

dowl:add - adds an OWL thing to the ontology.

If E is `dowl:add` and $e_I = MI_E(S_E(E))$ and $e_J = MJ_E(S_E(E))$

then the object exists only in the pre change ontology:

$$e_I = \emptyset$$

$$e_J \in R_J$$

Annotation Implications - the object must be added to the ontology:

$$(\text{owl:Thing}, T_I(e_I), \emptyset) \in A_A$$

Invalidating conditions - none.

dowl:rename - changes the URI reference of an object, it does not change the denotation of the object.

If E is `dowl:rename` and $e_I = MI_E(S_E(E))$ and $e_J = MJ_E(S_E(E))$

then - the original object is in the pre-change and the renamed object is in the changed ontology, the pre-change object is equivalent to the changed object.

$$e_I \in R_I \text{ and } e_I \ni R_J$$

$$e_J \in R_J \text{ and } e_J \ni R_I$$

$$R_I(e_I) \equiv R_J(e_J)$$

Annotation Implications - the object's name must be changed (this is the only reannotation that can be made on a proof):

$$\forall (x \mid x = R_I(e_I) \text{ and } x \in R_A) \langle T_I(x), e_J \rangle \in A_C$$

Invalidating conditions - None.

dowl:broaden - broadens the concept of an entity so that it completely encloses the pre-change entity.

If E is `dowl:broaden` and $e_I = MI_E(S_E(E))$

then F_P s defined by the `dowl:mapping`, `dowl:mappedTo`, and `dowl:mappedFrom` operators:

$$\forall (P, x, y, z \mid \langle P, z \rangle \in EXT_I(S_I(\text{dowl:Mapping}))$$

$$\text{and } \langle z, x \rangle \in EXT_I(S_I(\text{dowl:MappedFrom}))$$

$$\text{and } \langle z, y \rangle \in EXT_I(S_I(\text{dowl:MappedTo}))$$

$$\langle x, y \rangle \in F_P$$

If x is an instance of the broadened class in the pre-change ontology, x in an instance of the class in the changed ontology:

$$\forall (x \mid x \in CEXT_I(e_I) \text{ and } e_I \in CEXT_I(\text{owl:Class})) \ x \in CEXT_J(e_I)$$

Replace the values for datatype properties with the values specified in the mapping defined above:

$$\forall (x, p, z \mid x \in CEXT_I(e_I)$$

$$\text{and } p \in CEXT_I(S_I(\text{dowl:datatypeProperty}))$$

$$\text{and } \langle x, z \rangle \in EXT_I(y))$$

$$\langle x, F_{S_I(p)}(z) \rangle \in EXT_J(y)$$

Annotation Implications - Replace values for datatype properties with the values specified in the mapping defined above:

$$\forall (x, p, z \mid x \in CEXT_I(e_I)$$

$$\text{and } p \in CEXT_I(S_I(\text{dowl:datatypeProperty}))$$

$$\text{and } \langle x, z \rangle \in EXT_I(y)) \ \langle x, F_{S_I(p)}(z) \rangle \in A_C$$

Invalidating Conditions

None.

dowl:isNotSameAs - asserts that an object is not the same as another object.

If E is `dowl:notSameAs` and $e_I = MI_E(S_E(E))$ and $e_J = MJ_E(S_E(E))$

then the first object is not the same as, same class as, or same property as the second object:

$$\langle e_I, e_J \rangle \notin EXT_J(S_J(\text{owl:sameAs}))$$

$$\langle e_I, e_J \rangle \notin EXT_J(S_J(\text{owl:sameClassAs}))$$

$$\langle e_I, e_J \rangle \notin EXT_J(S_J(\text{owl:samePropertyAs}))$$

Annotation Implications - Any of the statements asserting equivalency between the two objects are removed:

$$(e_I, \text{owl:sameAs}, e_J) \in A_R$$

$$(e_J, \text{owl:sameAs}, e_I) \in A_R$$

$$(e_I, \text{owl:sameClassAs}, e_J) \in A_R$$

$$(e_J, \text{owl:sameClassAs}, e_I) \in A_R$$

$$(e_I, \text{owl:samePropertyAs}, e_J) \in A_R$$

$$(e_J, \text{owl:samePropertyAs}, e_I) \in A_R$$

Invalidating conditions - Sameness is transitive thus any statements which imply that e_I is the same as e_J invalidate the ontology:

$$\exists (x, Y, Z \mid \langle e_I, x \rangle \in EXT_J(S_J(Y)) \text{ and } \langle x, e_J \rangle \in EXT_J(S_J(Z))$$

$$\text{and } Y \in \{\text{owl:sameAs}, \text{owl:sameClassAs}, \text{owl:samePropertyAs}\}$$

$$\text{and } Z \in \{\text{owl:sameAs}, \text{owl:sameClassAs}, \text{owl:samePropertyAs}\})$$

dowl:undefineAsObjectProperty - removes the element from the class extension of ObjectProperty.

If E is `dowl:undefineAsObjectProperty` and X is `owl:ObjectProperty` and $e_I = MI_E(S_E(E))$

then the element is not an instance of object property:

$$e_I \notin CEXT_I(X)$$

Annotation Implications - replace any statement asserting that the element is an object property with a statement asserting that the element is an `owl:Thing`:

$$\forall(x \mid x \in CEXT_I(e_I) \text{ and } x \in EXT_A(X)) \langle T_I(x), \text{owl:Thing} \rangle \in A_C.$$

Invalidating conditions - if the element relates objects to objects and is not also an annotation property:

$$\begin{aligned} &\exists(\langle x, y \rangle \mid x \in CEXT_I(S_I(\text{owl:Thing})) \\ &\quad \text{and } y \in CEXT_I(S_I(\text{owl:Thing})) \\ &\quad \text{and } \langle x, y \rangle \in EXT_I(e_I)) \\ &\quad \text{and } \langle e_I, S_I(\text{owl:AnnotationProperty}) \rangle \\ &\quad \in EXT_I(S_I(\text{rdf:Type})) \end{aligned}$$

If a **proof** relies on the element being an object property:

$$\langle e_I, S_I(X) \rangle \in EXT_\rho(S_I(\text{rdf:Type}))$$

6. Conclusion

In this paper we have shown how the semantics of ontology change can be formally described. Our formalism is set in the context of the OWL ontology language and our theory is an extension of the RDF compatible model theoretic syntax and semantics for OWL. Using our extensions to the model theory in conjunction with an OWL based language for describing ontology change we described the semantics of a variety of OWL ontology changes derived from a generic model of ontology change. The key element of these descriptions is the formal description of the ramifications of the ontology change, that is the actions that need to be taken in order for the ontologies dependant on the changed ontology to be consistent with the changed ontology.

The next step in this work is the development of a system which can interpret ontology changes described in our DOWL language, and using the formal theory as a guide, automatically update ontologies dependant on the changed ontology. Such a system will form the core of a complete ontology change management solution.

References

- [1] John Avery and John Yearwood. Dowl: A dynamic ontology language. In *"IADIS WWW/Internet 2003, Faro, Portugal, 2003.*
- [2] John Avery and John Yearwood. Supporting evolving ontologies by capturing the semantics of change. In *"Ausweb '04, Gold Coast, Australia", 2004.*
- [3] E Franconi, F Grandi, and F Mandreoli. A semantic approach for schema evolution and versioning in object-oriented databases. In *"Computational Logic 2000, number 1861 in Lecture Notes in Computer Science",* pages 1048–1062, 2000.
- [4] Michel Klein. Supporting evolving ontologies on the web. In *Proceedings of the EDBT 2002 PhD Workshop,* March 2002.
- [5] Michel Klein. Change management for distributed ontologies. In *"PhD Thesis, Vrije Universiteit Amsterdam", 2004.*
- [6] N. F. Noy and M. A. Musen. Ontology versioning as an element of an ontology-evolution framework. In *IEEE Intelligent Systems,* 2003.
- [7] W3C. Resource description framework (rdf) model and syntax specification. In Ora Lassila and Ralph R. Swick, editors, *<http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>,* 1999.
- [8] W3C. Owl web ontology language guide. In *<http://www.w3.org/TR/2004/REC-owl-guide-20040210/>,* 2004.
- [9] W3C. Owl web ontology language semantics and abstract syntax. In *<http://www.w3.org/TR/2004/REC-owl-semantics-20040210/>,* 2004.