Chapter 6

AN OWL-DL IMPLEMENTATION OF GOLD

An Ontology for the Semantic Web

Scott Farrar[*] and D. Terence Langendoen[**]

[*] *Department of Linguistics*
University of Washington

[**] *Division of Information & Intelligent Systems*
National Science Foundation[*]

Abstract An OWL-DL implementation of the General Ontology for Linguistic Description (GOLD) is presented with relevant examples of axioms given throughout. As background, an introduction to Description Logic is presented using examples from linguistics and with particular attention to $\mathcal{SHOIN}(\mathbf{D})$, the logic which most closely relates to OWL-DL. The types of axioms used to develop an ontology in OWL-DL are explained. In addition, a domain independent methodology is given for creating description-logic based ontologies of any kind, not just those for linguistics. Using the $\mathcal{SHOIN}(\mathbf{D})$ notation, the methodology is demonstrated for the linguistics domain with particular attention given to illustrating the use of each type of axiom. Finally, the relevant issues and limitations to linguistic modeling in OWL-DL are discussed.

Keywords: OWL-DL, Description Logic, ontology, language description

Introduction

The General Ontology for Linguistic Description (GOLD) is an ontological theory for the domain of linguistics. Motivations for an ontolog-

[*]This material is based in part upon work supported while Langendoen was serving at the National Science Foundation. Any opinion and conclusions are those of the authors and do not necessarily reflect the views of the National Science Foundation.

ical theory for linguistics, including GOLD, have been given elsewhere (e.g. Farrar and Langendoen, 2003 and Farrar, 2007). Minimally, an ontological theory specifies the entities of interest in a given domain. Those entities include classes and their instances along with the relations that hold among those instances. Lightweight ontologies stop there, by providing an enumeration of the classes and a limited number of relations, usually enough to arrange the classes in a taxonomy. A more comprehensive ontology—some do not refer to light-weight ontologies as ontologies at all—places many more restrictions on the entities in the domain and can serve to facilitate automated reasoning. The logical sentences used to make explicit assertions about the base entities are referred to as **axioms**. That axioms play a crucial role in the design of formal ontologies is well known in the knowledge engineering literature (Niles and Pease, 2001, Masolo et al., 2003, among others).

The goal of the current work, then, is to demonstrate how to axiomatize one such ontology, GOLD.[1] The point of the current chapter is not only to present a particular aspect of an ontology for linguistics, but also to explore the limitations of OWL-DL (Smith et al., 2004) with respect to ontological modeling of the linguistics domain. This chapter could be written for any particular domain of inquiry, not just linguistics, to which an ontological theory is applied. However, it is the hope that this chapter, through its use of linguistic examples, will be particularly relevant for linguists who are interested in modeling one aspect of language or another.

As background for this task, Section 1 includes an introduction to knowledge engineering and ontologies. This section details the class of logical formalisms known as **description logic** (DL) that has led to the creation of OWL-DL (Horrocks et al., 2003). Included is a discussion of the standard notation for generic DLs and an in-depth look at $\mathcal{SHOIN}(\mathbf{D})$, the description logic which most closely relates to the final OWL-DL implementation. The use of DL notation in this chapter is justified for reasons of brevity, as the XML syntax for OWL-DL is too verbose to present in running text. In addition, a step-by-step methodology for creating an ontological theory is given in Section 2. Next in Section 3, the methodology for ontology creation is demonstrated using $\mathcal{SHOIN}(\mathbf{D})$ axioms. Finally in Section 4, the limitation to ontological modeling in OWL-DL are presented with a discussion on why OWL-DL is a suitable modeling language for certain reasoning tasks.

1. Background

In order to understand how GOLD is to be axiomatized, this section presents the relevant background. After a brief introduction to knowledge engineering and ontologies, we give a detailed discussion of the main formalism to be used throughout the paper, namely that of description logic. After an introduction to description logic in general, we give an overview of $\mathcal{SHOIN}(\mathbf{D})$, the description logic which most closely resembles that of OWL-DL. We then present a methodology whereby a knowledge base can be instantiated by using a description logic.

1.1 Knowledge engineering and ontologies

Knowledge engineering is the task of representing the knowledge of a particular domain in a machine readable format. For a particular knowledge engineering task, the formal language used to represent the knowledge often has far-reaching effects as to what kinds of domain knowledge can be captured by the representation. Furthermore, the product of knowledge engineering, the knowledge base, can be used in conjunction with automated reasoning tools to produce new knowledge, to prove the consistency of existing knowledge, and to enhance search within the knowledge base. The central assumptions in a knowledge base are captured in the **ontological theory**, or the set of statements that make up the essential knowledge of the domain, in other words, the knowledge that must always hold if the theory is to be coherent. We refer to such sets of statements as simply the *ontology*.

The statements included in the ontology hold according to a particular **conceptualization** of the domain. A conceptualization is an abstract, simplified view of the world (Gruber, 1993). Due to the complexity of any real-world domain, a conceptualization is necessarily a simplified approximation of reality. Still, that the conceptualization is approximate does not preclude it from being useful. A major issue in the modeling of the linguistics domain is that various linguistic theories adopt different and often incompatible conceptualizations. In fact, from the standpoint of the ontologist, the aim of science may be cast as the search for the ideal conceptualization. But admittedly, what is meant by "ideal" can vary according to the task at hand. For our task, achieving interoperability over a broad spectrum of language descriptions, we require only that our conceptualization be rich enough to account for differences in various linguistic descriptions. The nature of this task relaxes some of the requirements on the ontological theory. An ontology for all of linguistics is, at this point, unachievable and would require deep consensus as to how language is conceptualized. Still, we will undoubtedly come across descriptions that are incompatible with one another due to different theoretical assumptions, in other words, disparate conceptualizations.

1.2 Description logic

The task of ontology building requires the use of logic as a means of axiomatization. First-order logic (FOL), for instance, is one well understood language for this task and is often employed, in one form or another, for this purpose; see the ontologies of SUMO (Niles and Pease, 2001) and DOLCE (Masolo et al., 2003). An alternative to FOL in the design of knowledge-based systems is the class of logics known collectively as **Description Logics** (DLs) (Baader et al., 2003). A DL is a less expressive, but highly structured fragment of first-order logic. Using a DL buys improved computational tractability but at the cost of expressivity. This means that algorithms for working with DLs will be fast, but that expressing certain concepts in a DL might not be possible. This section gives an introduction to this class of logics by discussing some of the key properties of DLs illustrated by examples in typical DL notation. Furthermore, we limit our discussion here to the linguistics domain.

1.2.1 Basic notions and terminology. A description logic is a formal logic in the strict sense. That is, it has a formal syntax which specifies how to construct well-formed sentences and a formal semantics which relates those sentences to a model. A description logic, as with all formal logics, has an associated proof theory, or a system of how certain entailments follow from a set of sentences. The focus of this section is mainly on the syntactic operations of description logic, but supplemented with an informal discussion of semantics. For a full account of the semantics of description logic, see Baader et al., 2003.

Whereas the predicates in FOL have equal ontological status, those in a DL come in two sorts: **concepts** and **roles**. Concepts in a DL correspond to unary predicates, while DL roles are used in place of binary predicates. What are referred to as constants in a first-order logic are referred to as **individuals** in DL. Intuitively a concept represents a category or kind in the domain being modeled. A concept is a universal notion and can be instantiated by individuals. The relation of **instantiation** holds between concepts and individuals, making an individual an instance of some concept (Nardi and Brachman, 2003). Individuals are disjoint from concepts and cannot be instantiated or related by the subsumption relation. A role is a binary relation between individuals. Description logic by definition has only binary relations and, thus, relations of higher arity (e.g. ternary relations) are disallowed. The terms *concept* and *role* show that the origins of description logic lie in the early work on knowledge representation, particularly on **frame-based** languages (Baader et al., 2003). In such languages, information is gathered together into structures called *frames* (structured objects), each particular type of which admits a specified set of possible attributes related by slots (roles).

The terms *concept*, *individual*, and *role* are particular to the body of literature concerning description logics. More general works in ontology and knowledge engineering use *class*, *instance*, and (binary) *predicate*, instead of the DL-specific terms. In the language of OWL-DL, *property* is—confusingly—used in place of binary relation. Though the current work is meant to guide the reader in constructing OWL ontologies, we use DL terminology throughout, mainly for the sake of consistency since logical formulas are given in DL notation.

Within a description logic system, concepts and roles are separated from individuals by partitioning the **knowledge base** into a **TBox** (short for *terminology box*) and an **ABox** (short for *assertion box*, in the sense that assertions are made about a given terminology.). The TBox consists of axioms about the domain in general in the form of logical sentences, while the ABox consists of facts about individuals. A description logic knowledge base KB may be defined minimally as the tuple consisting of a TBox T and an ABox A, i.e. $KB = \langle T, A \rangle$, where T is the union of the set of concepts with the set of roles in the domain, and A is the set of individuals in the domain; furthermore, the TBox also contains axioms relating to concepts and roles, while the ABox contains axioms relating to individuals.

Description logic can be used to represent much more than just basic concepts and individuals. Complex, non-atomic concepts can be specified through logical statements. Statements in a DL differ considerably from those in standard FOL. Moreover, statements in a DL are expressed at the level of predicates, i.e., there are no variables. Thus, axiom (1) gives an expression in a DL.

$$\mathsf{InflectedUnit} \equiv \mathsf{GrammarUnit} \sqcap \exists\, \mathsf{hasConstituent.InflectionalUnit} \qquad (1)$$

This can be glossed as: "The class InflectedUnit is defined as the intersection of the class of GrammarUnit and any class having at least one hasConstituent role whose value is restricted to the class InflectionalUnit." (See Section 3.1.2 for a further explanation of this axiom.) Statements in a description logic are therefore formulas containing predicates, technically with one free-variable, but omitted in the syntax. Predicates in a DL represent concepts and roles. Concepts are either **atomic**, i.e. those identified by name and may or may not be defined,[2] or **complex**, i.e.

those derived from atomic concepts using a set of **constructors**. The supported concept and role constructors in a particular DL determine its expressive power (Horrocks et al., 2003, p. 6). Thus, the constructors are used to derive well-formed formulas. In the following we have listed some examples of very common constructors in DLs with notes about their respective semantics and how they could be used in an ontology for linguistics. Furthermore, these constructors and others are used to create right-hand side expressions that define anonymous concepts. Any of the expressions below could be placed with a named concept on the left and related with either $\equiv$ or $\sqsubseteq$ (see Section 1.2.2 for definition of these symbols).

conjunction ($\sqcap$):

$$\mathsf{AfricanLanguage} \sqcap \mathsf{EndangeredLanguage} \tag{2}$$

Expression (2) can be glossed as "those individuals which are shared between the concepts AfricanLanguage and EndangeredLanguage". Conjunction is interpreted as the intersection of sets of individuals.

disjunction ($\sqcup$):

$$\mathsf{TenseFeature} \sqcup \mathsf{AspectFeature} \tag{3}$$

Expression (3) can be glossed as "the individuals that either belong to the concept TenseFeature or AspectFeature". Disjunction is interpreted as union of sets of individuals.

negation ($\neg$):

$$\neg\mathsf{PhonologicalUnit} \tag{4}$$

Expression (4) can be glossed as "the set of all individuals that are not instances of PhonologicalUnit". Negation is interpreted as the complement of a set of individuals with respect to the universal set of the domain.

existential quantifier ($\exists$):

$$\exists\, \mathsf{hasPart.GrammarUnit} \tag{5}$$

Expression (5) can be glossed as "the set of individuals each of which has some member of GrammarUnit as its part." The expression does not limit things other than members of GrammarUnit from being parts. The fact that other entities could be members of $\mathsf{GrammarUnit}$ is because of the open-world assumption built into the DL, namely that the domain is not assumed to be complete unless explicitly stated. To make the above description of quantification clear, the following serves to compare a simple DL formula (6) with the equivalent in standard FOL (7).

$$\exists\,\mathsf{R.C} \tag{6}$$

$$\{x|\exists y\, R(x,y) \wedge C(y)\} \tag{7}$$

universal quantification (∀):

$$\forall\,\mathsf{hasFeature.MorphosyntacticFeature} \tag{8}$$

Expression (8) can be glossed as "the set of individuals whose features are *only* individuals of $\mathsf{MorphosyntacticFeature}$". Universal quantification restricts all roles of some concept to be value-restricted by concepts of a certain type. Universal quantification does not ensure that there will be a role that satisfies the condition, but if there are such roles, their ranges have to be restricted to the given type. Again, the following serves to compare a simple DL formula (9) with the equivalent in standard FOL (10).

$$\forall\,\mathsf{R.C} \tag{9}$$

$$\{x|\forall y\, R(x,y) \rightarrow C(y)\} \tag{10}$$

1.2.2 Beyond the basics. With a brief introduction to description logic out of the way, we now focus on a variety of description logic that is the most useful in constructing an ontology using OWL-DL, namely a description logic called $\mathcal{SHOIN}(\mathbf{D})$. First off, $\mathcal{SHOIN}(\mathbf{D})$ is a notational variant of the OWL-DL language and is derived from a family of description logics referred to as the $\mathcal{SHOIN}(\mathbf{D})$ family. As in the naming of other DLs, the expressive power of $\mathcal{SHOIN}(\mathbf{D})$ is reflected in its name. For example, the S is due to the family's relationship to the modal logic S4 (Horrocks et al., 1999). The other components of the name are be described as follows: H means that role hierarchies are included; O means that individuals are included; I means that inverse roles are allowed; N means that number restrictions are allowed;

and *(D)* means the optional inclusion of concrete data types. In order to encode knowledge in $\mathcal{SHOIN}(\mathbf{D})$, and eventually in OWL-DL, an understanding of the allowed constructors for $\mathcal{SHOIN}(\mathbf{D})$ is necessary. These, along with the corresponding OWL-DL constructors, are listed in Table (6.1).[3]

Constructor	$\mathcal{SHOIN}(\mathbf{D})$	**OWL-DL**
conjunction	$C_1 \sqcap C_2$	unionOf(C_1, C_2)
disjunction	$C_1 \sqcup C_2$	intersectionOf(C_1, C_2)
negation	$\neg C_1$	complementOf(C)
oneOf	$\{o_1, ..., o_n\}$	oneOf(o_1,...,o_n)
exists restriction	$\exists R.C$	someValuesFrom(C); onProperty(R)
value restriction	$\forall R.C$	allValuesFrom(C); onProperty(R)
atleast restriction	$\geqslant nR$	minCardinality(n); onProperty(R)
atmost restriction	$\leqslant nR$	maxCardinality(n); onProperty(R)
datatype exists	$\exists R.D$	someValuesFrom(D); onProperty(R)
datatype value	$\forall R.D$	allValuesFrom(D); onProperty(R)
datatype atleast	$\geqslant nR$	minCardinality(n); onProperty(R)
datatype atmost	$\leqslant nR$	maxCardinality(n); onProperty(R)
datatype oneOf	$\{v_1, ..., v_n\}$	oneOf(v_1,...,v_n)

Table 6.1. A comparison of $\mathcal{SHOIN}(\mathbf{D})$ and OWL-DL constructors.

An in-depth analysis of OWL-DL in relation to other description logics is given by Horrocks et al., 2003. Also, Horridge et al., 2004 is a the very helpful and practical guide to building OWL ontologies in the Protégé environment.

The basic constructors of a DL such as $\mathcal{SHOIN}(\mathbf{D})$ can be used on the right side of either the $\sqsubseteq$ or $\equiv$ symbol to create logical statements of various kinds (see below). The resulting logical statements are the axioms of a DL. A DL axiom may be defined as some restriction on a concept or role. It is an assertion of knowledge using the entities in the ontology; that is, an axiom holds *a priori* of any knowledge that is later generated using the ontology, at least in a monotonic knowledge system. Which kinds of axioms to include in an ontology is, of course, a major focus of ontological engineering. Axioms in a DL knowledge-based system can be classified according to what objects they describe (TBox or ABox entities) and according to whether or not they are definitional (necessary and sufficient). Based on these criteria, a taxonomy of the various sorts of DL axioms is given in Figure 6.1, and the remainder of this section explores each sort in turn.

Terminological axioms make statements about entities in the TBox, i.e., concepts and roles, not individuals. Terminological axioms for a

- Terminological (TBox) axioms
  - inclusions (necessary)
    - concept inclusions
    - specializations
  - equalities (necessary and sufficient)
    - concept equations
    - concept definitions
- Assertional (ABox) axioms
  - concept assertions
  - role assertions

Figure 6.1. A taxonomy of DL axioms sorts.

given concept can be classified as either necessary or as necessary *and* sufficient conditions to be included in that concept. Terminological axioms that give the necessary conditions for some concept to be included (subclassed) in another are called **inclusion** axioms. There are two types of inclusions.

The first type of inclusion is simply a **concept inclusion**. A concept inclusion has the abstract form $\mathsf{C} \sqsubseteq \mathsf{D}$. A concept inclusion states a necessary, but not sufficient, condition for membership in some concept. It can be read as “having property D is necessary for a TBox entity to be included in concept C, but this condition alone is not sufficient to conclude that the object is in concept C”. Both C and D can be arbitrary concept expressions. An example of this kind of axiom is given in (11).

$$\mathsf{Head} \sqcap \mathsf{Verb} \sqsubseteq \mathsf{MainVerb} \tag{11}$$

The second type of inclusion axiom is a **specialization**, which has the abstract form $\mathsf{A} \sqsubseteq \mathsf{C}$. This is very similar to that of the inclusion axioms, but specializations are different from concept inclusions because the left-hand side of a specialization must be atomic (hence A). The abstract form can be read as “having properties of concept C is necessary for an entity in order to be included in concept A”. Axiom (12) is an example of a specialization.

$$\mathsf{SemanticUnit} \sqsubseteq \mathsf{Abstract} \tag{12}$$

A specialization axiom is useful when some concept cannot be defined completely (Baader and Nutt, 2003, p. 58). Both sorts of inclusion axioms in a TBox can be viewed as a limited kind of logical implication (Nardi and Brachman, 2003, p. 18). Concept inclusion axioms are very important in the structure of the knowledge base as they are used to generate a taxonomy from a set of assertions in a TBox.

Another type of terminological axiom, those concerning concepts and roles, are equalities. A **concept equation** has the general form of $\mathsf{C} \equiv \mathsf{D}$ as in (13):

$$\mathsf{ContentBearingPhysical} \equiv \exists\, \mathsf{expresses.LinguisticSign} \tag{13}$$

This axiom simply states that a ContentBearingPhysical is defined as anything that realizes a LinguisticSign. A special kind of equation is a **concept definition** of the form $\mathsf{A} \equiv \mathsf{C}$ where the left-hand side is an atomic concept. A concept definition states the necessary and sufficient conditions that must hold in order for a TBox entity to be included in some other concept. Having property C is necessary and sufficient for a TBox entity to be included in concept A. Axiom (14) is an example of a concept definition:

$$\mathsf{Language} \equiv \mathsf{SpokenLanguage} \sqcup \mathsf{WrittenLanguage} \sqcup \mathsf{SignLanguage} \tag{14}$$

This can be glossed as "a language is either spoken, written, or signed; there is no other type of language". Axiom (14) is furthermore a **covering axiom**, as it guarantees that Language will only have 3 subclasses. Furthermore, a concept definition has the effect of introducing a symbolic name for some constructed concept into the TBox (Baader and Nutt, 2003, p. 55).

Finally, axioms that pertain only to individuals are called **assertional axioms**, hence the label ABox. Assertions can either pertain to concepts or roles. A concept assertion is of the form $\mathsf{C(I)}$, where C is some concept from the TBox and I is an individual. $\mathsf{C(I)}$ means that I is an instance of C. A linguistic example would be (15):[4]

$$\mathsf{Noun}(\textsc{Noun123}) \tag{15}$$

A **role assertion** is of the form $\mathsf{R(A,B)}$, where R is some role from the TBox and A and B are individuals. $\mathsf{R(A,B)}$ means that B is a filler of A for role R. An example of an assertional axiom is given in 16.

$$\mathsf{precedes}(\textsc{Noun123}, \textsc{Verb456}) \tag{16}$$

2. Methodology

The development of a knowledge base, including an ontology, is essentially a two-stage process. Figure 6.2 lists these steps specifically for a knowledge base for linguistics that incorporates GOLD as well as linguistic data. The following enumeration is in part adapted from Borgida and Brachman, 2003, p.379. The methodology given here supersedes an earlier version given in Farrar, 2007. The steps in the methodology are discussed in detail in the next section.

1 Design the TBox for the knowledge base.
   - (a) Classify entities as concept, role, or individual.
   - (b) Add concepts to TBox.
      - i Declare atomic concepts.
      - ii Define non-atomic (constructed) concepts.
      - iii Create concept taxonomy.
      - iv Partition the concept taxonomy.
   - (c) Add roles to TBox.
      - i Declare transitive and symmetric roles.
      - ii Declare inverse roles.
      - iii Declare functional roles.
      - iv Add domain and range restrictions to roles.
      - v Add cardinality restrictions to roles.
   - (d) Add other axioms to further refine concepts and roles.

2 Populate the ABox with individuals.
   - (a) Enumerate and classify each individual according to available concepts.
   - (b) Relate individuals via available roles in ontology.

3 Relate TBox and ABox.
   - (a) Create enumerated concepts.
   - (b) Relate individuals to concepts via roles.

Figure 6.2. The steps in creating a knowledge base in DL.

3. Linguistic modeling in OWL-DL

In this section we discuss the steps in creating a DL knowledge base for the descriptive linguistics domain. We use DL notation as introduced in Section 1.2, though the ultimate aim of this section is to act as a guide to creating such a knowledge base in OWL-DL. At issue are the specific kinds of axioms needed to express a wide variety of linguistic knowledge found in the domain. The structure of the current section mirrors the steps in methodology for creating a DL knowledge base listed in Figure 6.2.

3.1 Design the TBox for the knowledge base

This step includes developing the basic structure of the ontology. The most important task in creating any ontology is to properly enumerate the entities found in the domain. If the inventory is *ad hoc* or incomplete, then the resulting ontology will not be an accurate conceptualization. The key is to establish a rigid foundation such that later additions will not create problems for the overall theory. We refer to such a foundation as the **upper ontology**. For descriptive linguistics such an upper ontology contains the fundamental knowledge of structure, form, and meaning, that which is usually possessed by a well trained linguist. This, ideally, would include general knowledge that applies to any language or theoretical framework. Examples of general knowledge of this sort are given below:

- A verb is a part of speech.
- A verb can assign case.
- Gender can be semantically grounded.
- Linguistic expressions realize morphemes.

This kind of knowledge is typical of that represented in an ontology in knowledge based systems. The ontology provides the means of formalizing such expressions and defining them in a larger conceptual framework. For example, it provides the means of specifying how a spoken linguistic expression is related to the printed form of a writing system, or how Tense is defined in terms of a temporal calculus.

Of the most fundamental entities that occur in the linguistics domain are the linguistic expressions themselves. The basic entities here are orthographic expressions, spoken expressions, and signed expressions. Other than such concepts that are physical in nature, those occupying time and space, there are the abstract concepts such as the units of grammatical structure, and meaning. As presented in Farrar, 2007, these three types of entities are unified under the concept of Sign via a set of relations. From these three fundamental types, the basic units of linguistic analysis can be derived, including concepts such as Glyph, Phoneme, SyntacticCategory, SemanticUnit, etc.

Next, there are the entities that relate the fundamental units to one another. For instance, two expressions can be related via precedence in time and/or space, but also via dominance relations as in grammatical structure. The mereology of such units is a necessary component in the ontology, that is, how units are composed of other units. Consider the example of sound structure. There are the basic phonological entities, in general, PhonologicalUnit, including the concept of Phoneme. Larger phonological units include the PhonologicalWord. Parts of the phonological unit include the Mora. Each level of linguistic analysis with have its own unit types, relations, and theory, in short, its own mereology.

Next, there are more specific entities that may be considered as part of the overall upper ontology, for instance, the various *features* associated with the fundamental units. Features can be phonological, morphosyntactic, syntactic or semantic: MorphosyntacticFeature, SyntacticFeature, PhonologicalFeature, and SemanticFeature. Depending on the level of granularity, the various kinds of features may be divided into subgroups. For instance, TenseFeature and NumberFeature are both kinds of MorphosyntacticFeature.

Finally, there are the various structuring devices used in linguistic analysis. In general, we refer to these as **linguistic data types**. There are several fundamental types, including Lexicon, GlossedText, PhonologicalParadigm, FeatureStructure, StructuralDescription, etc. Each of these data types has its own mereology, e.g. FeatureStructure which is the pairing of a feature name and a feature value.

3.1.1 Classify entities as concept, role, or individual. As discussed in Section 1.2.1, a concept in DL represents a category or kind in the domain being modeled. A concept is a universal notion and can be instantiated by individuals. Concepts in a DL, then, are classes of individuals. The binary relations that hold among various individuals are known as roles. The next crucial task in creating the ontology is to decide to which sort each entity in the domain belongs.

In any DL , the most basic distinction is between concepts and roles. Such a decision is perhaps the most intuitive of all modeling decision. This is reflected in how entities are named. That is, it often possible to simply assign entities to either concept or role based on whether they are named using nouns or verbs respectively, at least in English. For instance, consider the notion feature and the relating of a feature to its value. We may refer to feature simply as Feature, a noun in English and hence a concept. We may refer its having a value as hasValue, as in the DL literature, in which there is a strong tendency to name roles using the word *has* combined with the concept that acts as the range, thus, hasValue, hasPart, hasHead, etc. There are of course problematic cases pertaining to the distinction between concepts and roles.

Consider the example where a verb is said to assign case. On the one hand, one could posit the role of assignsCase and include Verb as the domain of the role. On the other hand, one could use the concept of CaseAssigner, say that pertained to verbs, determiners, etc. to create the complex concept of CaseAssigningVerb. On the one hand "things" have an unchanging essence. A stone is a still a stone even when it is used as a doorstop or a weapon. On the other hand, roles that things play can change depending on the context. At one moment, John may student of guitar, while at another, he may be a professor of philosophy. In a DL system, it is preferable to limit the number of concepts when possible by using roles that help to extend concepts to form others. For instance, we may enumerate several atomic syntactic categories, e.g. Noun, Verb, Determiner, and then use the role assignsCase to compose complex concepts when needed, such as CaseAssigningVerb, a particular type of case assigner that happens to be a verb. It would be even simpler and more advantageous in a DL to simply use the role assigns. In this way, one could enumerate various categories such as Case, Gender, and Number and use assigns to derive concepts such as GenderAssigningNoun and CaseAssigningVerb.

Next, there is the issue of whether an entity is a concept or an individual. In some cases, the distinction is quite clear. Consider the notions of *Germanic* versus *standard German*. Since we know that there is more than one type of Germanic language, we can feel assured that Germanic is a concept. Likewise, since we know that there is usually only one variety referred to as standard German, we might propose HochDeutsch as that individual, that is, an instance of Germanic, as in (17):

$$\mathsf{Germanic}(\textsc{HochDeutsch}) \tag{17}$$

Problems arise when the domain is conceptualized differently, for instance, when entities such as *Germanic* are treated as individuals, for instance, when reasoning about specific groups of languages is needed. As shown in (18), one could introduce the concept of LanguageFamily such that its instances included the individuals such as Germanic, Tibetan, Bantu, etc.

$$\mathsf{LanguageFamily}(\textsc{Germanic}), \mathsf{LanguageFamily}(\textsc{Tibetan}), \ldots \tag{18}$$

How then does the individual HochDeutsch relate to individual Germanic? Statement (17) is no longer allowed. In OWL-DL it is not possible for a concept to be both a class and an instance. One solution to the problem is to introduce another type of role, for instance inFamily, such that LanguageFamily is a concept with GERMANIC as an instance, and HOCHDEUTSCH relates to GERMAN via the inFamily role, summed up in axioms (19–21):

$$\mathsf{LanguageFamily}(\textsc{Germanic}) \tag{19}$$

$$\mathsf{LanguageVariety}(\textsc{HochDeutsch}) \tag{20}$$

$$\mathsf{inFamily}(\textsc{HochDeutsch}, \textsc{Germanic}) \tag{21}$$

This solution favors the treatment of Germanic as an individual. Note that with this particular conceptualization, there is no need for the concept GermanicLanguage as a subclass of LanguageVariety. Germanic takes its place. Our treatment easily allows for competing classifications of languages since different ABoxes (corresponding to different classification schemes) could be developed from the same TBox (corresponding to the non-controversial knowledge of the field, namely that there are families and varieties, with no specific classification implied).

3.1.2 Add concepts to TBox. Once entities have been classified as concepts, they can now be added to the TBox. There are (potentially) two kinds of concepts and each is treated differently. First, there are concepts that are assumed to exist in the absence of any definitional axioms. These are known as **atomic concepts** and are entered into the TBox using unique names. Next, there may be concepts that are defined in terms of other concepts, referred to as **non-atomic**. For instance, (22) shows an example of a defined concept.

$$\mathsf{InflectedUnit} \equiv \mathsf{GrammarUnit} \sqcap \exists\, \mathsf{hasConstituent}.\mathsf{InflectionalUnit} \tag{22}$$

Thus, an inflected unit is defined in terms of grammar unit and inflectional unit. Specifically, the axiom states that it is not possible for some individual to be an inflected unit with having some inflectional unit as one of its constituents.

With the entities enumerated and classified as one of the three DL sorts, it is now possible to add structure to the ontology by classifying them according to the subsumption relation. Recall that subsumption is a built-in partial-ordering relation—it is reflexive, transitive and antisymmetric—and is used to form concept and role taxonomies. In a DL, subsumption is a type of inclusion axiom in the form of $A \sqsubseteq B$, where

A is subsumed by B. The crucial point at this step is not to misinterpret the intended meaning of subsumption. One common mistake is to interpret subsumption as the part-whole relation. For instance, consider the various phonological concepts: PhonologicalWord, Syllable, Foot and Mora. These can be related via the part-whole relation to form a mereology, such that a Foot is part of a PhonologicalWord, a Syllable is part of a Foot (of course by transitivity, a Syllable is also part of a PhonologicalWord), and a Mora is part of a Syllable (or more precisely a part of a Coda, which is part of a Syllable). It would be a mistake to use subsumption in this manner. Instead, there is some concept, call it PhonologicalUnit, that subsumes all the aforementioned phonological concepts:

$$\mathsf{PhonologicalWord} \sqsubseteq \mathsf{PhonologicalUnit} \tag{23}$$

$$\mathsf{Syllable} \sqsubseteq \mathsf{PhonologicalUnit} \tag{24}$$

$$\mathsf{Foot} \sqsubseteq \mathsf{PhonologicalUnit} \tag{25}$$

$$\mathsf{Mora} \sqsubseteq \mathsf{PhonologicalUnit} \tag{26}$$

The mistake arises from the fact that 'is part of', like 'is subsumed by', is a partial ordering, but is not reducible to it. Every PhonologicalWord has parts, each of which is a MetricalFoot, as expressed in (27).

$$\mathsf{PhonologicalWord} \sqsubseteq \exists\, \mathsf{hasPart.MetricalFoot} \tag{27}$$

This is distinct from saying that PhonologicalWord subsumes MetricalFoot. If every MetricalFoot is a PhonologicalWord, then it is correct to say that PhonologicalWord subsumes MetricalFoot, but that is very different from saying that every PhonologicalWord is made up of at least one MetricalFoot.

Partition the concept taxonomy. The next step is to create partitions in the concept taxonomy. A partition ensures that two or more concepts never share individuals; such concepts are referred to as **disjoint**. If A and B are disjoint, then $\mathsf{A} \sqcap \mathsf{B} \equiv \bot$ must be true. In other words, the set formed by the intersection of two or more disjoint concepts will always be empty. Partitioning the domain is one way to speed up certain reasoning tasks, since in a DL concepts are assumed to overlap unless stated otherwise. As a linguistic example, consider the case of WrittenExpression versus SpokenExpression versus

SignedExpression. These three concepts may never share individuals since a given linguistic expression can never exist in more than one physical form at the same time. The spoken word "dog", the written word *dog*, and the sign for *dog* are all different entities. The three forms may be related in a regular way—the word *dog* definitely relates to its spoken counterpart—but no conceptualization allows for more than one simultaneous mode of being such as this. Thus, axioms (28–30) are necessary to handle the linguistic example:

$$\mathsf{SpokenExpression} \sqcap \mathsf{WrittenExpression} \equiv \bot \tag{28}$$

$$\mathsf{SpokenExpression} \sqcap \mathsf{SignedExpression} \equiv \bot \tag{29}$$

$$\mathsf{SignedExpression} \sqcap \mathsf{WrittenExpression} \equiv \bot \tag{30}$$

And finally, to ensure that there can be no other kind of Expression, we include the covering axiom (31):

$$\mathsf{Expression} \equiv \mathsf{SpokenExpression} \sqcup \mathsf{WrittenExpression} \sqcup \mathsf{SignedExpression} \tag{31}$$

3.1.3 Add roles to TBox. The next step is to add roles to the TBox. At this step, there are several issues to be addressed in order to properly characterize a role. First, roles in a DL can be classified as transitive. An example of a transitive role from the linguistics domain is the hasConstituent, holding among syntactic units. If A has constituent B and B has constituent C, then A also has constituent C.

It is also possible to declare roles as symmetric, as in translationOf since if A is a translation of B, presumably B is a translation of A. Likewise, many lexical relations are also symmetric: antonymOf, synonymOf, etc. The relation of agreement, denoted by agrees, is another example of a symmetric relation. An example of a role that is neither transitive nor symmetric would be the hasValue role that relates a feature to its value.

With a number of roles in place, the inverses of roles should now be declared. Inverse roles simply serve to express the opposite relation between two individuals. For instance, consider the relationship between two forms, one with inflection and one without. The role inflectedForm could be used to relate the base form to the inflected form, while baseForm could be used for the inverse. There is no need to state further axioms for the inverse, since inflectedForm is already defined. Other examples include linear ordering relations in morphology such as follows and precedes.

It is possible to add domain and range restrictions to roles thereby restricting a given role to taking particular concepts as its domain and particular concepts as its range. Axiom (32), for instance, limits the range of hasHead to only individuals of type SyntacticWord.

$$\top \sqsubseteq \forall\, \mathsf{hasHead.SyntacticWord} \tag{32}$$

Note the use of $\top$. This symbol means that for anything in the knowledge base, the range of hasHead must be filled with a SyntacticWord. However, leaving the domain and range values open is often advantageous. Consider the part-whole relations with respect to morphosyntactic structure. It would be useful to have a single relation hasConstituent that could hold between any complex structure and its constituent. We might want this relation to pertain to both syntactic units and morphological units. If we declared strict domain and range constraints then we would need a different relation in each case, something like hasMorphConstituent and hasSynConstituent. But by leaving the domain and range constraints open, we can get by with using hasConstituent. Other axioms such as (33) may be added to constrain its use:

$$\mathsf{Phrase} \sqsubseteq \exists\, \mathsf{hasConstituent.SyntacticWord} \tag{33}$$

Cardinality restrictions place restrictions on the number of particular kinds of relationships a concept may participate in. For instance, language endangerment could be defined (albeit simplistically) as any language that has no more than 2,000 speakers.

$$\mathsf{EndangeredLanguage} \equiv \mathsf{Language} \sqcap \leqslant 2000\, \mathsf{hasSpeaker} \tag{34}$$

Roles may also be arranged into a taxonomy to add structure to the TBox. This is useful when there are several roles, for example, that share the same range. Consider the various kinds of syntactic roles that hold between constituents and the main clause: hasPredicate, hasSubject, hasObject, etc. We might say that there is a single role, say syntacticRole, that subsumes all of these:

$$\mathsf{hasPredicate} \sqsubseteq \mathsf{syntacticRole} \tag{35}$$

$$\mathsf{hasSubject} \sqsubseteq \mathsf{syntacticRole} \tag{36}$$

$$\mathsf{hasObject} \sqsubseteq \mathsf{syntacticRole} \tag{37}$$

The result of adding the above axioms is a role taxonomy, must like the concept taxonomy discussed earlier. In practice, role taxonomies will not be as detailed as those for concepts.

3.1.4 Add other axioms to further refine concepts and roles.

The next step is to establish role restrictions on concepts, that is, to assert how individuals of particular concepts are related via roles. Such relations can be asserted using either the existential or the universal role restriction. An example using the existential is that any morphosyntactic feature must have at least one value that is a morphosyntactic value.

$$\mathsf{MorphosyntacticFeature} \sqsubseteq \exists\, \mathsf{hasValue.MorphosyntacticValue} \tag{38}$$

In order to ensure that some feature does not have any other type as its value, then the right-hand side of (38) must be augmented as in (39):

$$\mathsf{MorphosyntacticFeature} \sqsubseteq \ldots \sqcap \forall\, \mathsf{hasValue.MorphosyntacticValue} \tag{39}$$

Thus, (39) combines the existential with the universal to achieve a tighter restriction and can be glossed as “a morphosyntactic feature must have at least one morphosyntactic value as its value (the existential) and only morphosyntactic values can be related to morphosyntactic feature via the hasValue role (the universal)”. If the universal were used by itself, then it would be possible for a feature *not* to have a value. The existential-universal combination, then, is a common design pattern used in ontology engineering with OWL-DL (Horridge et al., 2004).

The part-whole relationship has already been introduced with reference to mereology. Part-whole relationships apply in a straight-forward manner to phonological and morphosyntactic structure. For instance, constituency structure can be spelled out by using axioms such as (40):

$$\mathsf{SyntacticPhrase} \sqsubseteq \exists\, \mathsf{hasContituent.SyntacticWord} \tag{40}$$

This axiom states simply that syntactic phrases must have at least one syntactic word as a constituent.

3.2 Populate the ABox with individuals

The next step is to instantiate the various concepts and thereby populate the knowledge base, in fact the ABox, with individuals. This step includes enumerating the individuals, sorting them, and finally asserting knowledge about each individual.

3.2.1 Enumerate and classify each individual according to available concepts. This step concerns adding concrete data to the schema knowledge contained in the knowledge base. An obvious example is to add all the known language varieties to the concept of LanguageVariety. A single statement such as (41) is required for each introduction and classification of an individual.

$$\mathsf{LanguageVariety}(\textsc{OldEnglish}) \tag{41}$$

The task of determining exactly what are the language varieties that exists is of course contentious. In general, the TBox should contain universal (or widely accepted) knowledge, while the ABox should be dedicated to that which may be contested. In general, the TBox/ABox separation represents the split between general linguistic knowledge and that pertaining to individual languages, the latter of which is usually more contested. For example, although the fact that Hopi has an ImperfectiveAspect and that English and Greek both have a PastTense constitute linguistic knowledge (perhaps widely held, general knowledge), this kind of knowledge can be differentiated, as it only pertains to specific languages and, thus, is better placed in the ABox as individuals. The drawback is that this would lead to an explosion in the number of individuals required for even an ontology that included a relatively small number of languages: HopiPastTense, EnglishPastTense, GreekPastTense, . . .

3.2.2 Relate individuals via available roles in ontology. With the axiomatization in the TBox in place, it is a relatively straightforward procedure to assert knowledge about individuals, for instance to relate each individual language to the country where it is spoken:

$$\mathsf{spokenIn}(\textsc{Mandarin}, \textsc{China}) \tag{42}$$

$$\mathsf{spokenIn}(\textsc{Mandarin}, \textsc{Canada}) \tag{43}$$

3.3 Relate TBox and ABox

The final step in the methodology is to relate the TBox to the ABox by introducing mixed axioms that contain all three logical sorts: concepts, roles and individuals. Mixed axioms can be formed in a number of ways, for instance by relating a concept with the enumerated set of its individuals or by relating specific individuals to concepts using roles.

3.3.1 Create enumerated concepts. Enumerating a concepts individuals is useful when there is a need to tie a concept absolutely to a specific set of individuals, meaning that the set is not likely to change. For instance, when enumerating the values of some morphosyntactic system for a given language and these values are well-agreed upon, an enumerated concept could be used as follows:

$$\mathsf{TenseSystem} \equiv \{\textsc{DistantPast}, \textsc{HodiernalPast}, \textsc{Present}\} \tag{44}$$

This axiom fully defines TenseSystem by stating its necessary and sufficient conditions, namely, an enumeration of its individuals. Put another way, TenseSystem is equivalent to the set of the three given individuals.

3.3.2 Relate individuals to concepts via roles. Finally, it is possible to use an individual to place a narrow restriction on a given class. Assuming that the individual NULL represents the absence of phonetic material, a linguistic expression with no phonetic component, a zero morpheme could be defined as any morpheme that is expressed by the null element.

$$\mathsf{ZeroMorpheme} \equiv \exists\, \mathsf{expressedBy}.\{\textsc{Null}\} \sqcap \mathsf{Morpheme} \tag{45}$$

Secondly, individuals can be defined by relating them to concepts via roles.

$$\{\textsc{Verb123}\} \sqsubseteq \exists\, \mathsf{hasSyntRole.Agent} \tag{46}$$

The above axiom states that the specific verb VERB123 must have an agent as one of its syntactic roles.

4. Limitations and tool-related issues

4.1 The fundamental types

The first issue to be discussed is whether or not the fundamental predicate types—concepts (classes), roles (properties), and individuals—are adequate for linguistic modeling.[5] For concepts, there is little difference between a language such as OWL-DL and first-order ontology modeling languages like SUO-KIF. Concepts within an OWL-DL taxonomy behave in the expected way, structured by the partial-ordering relation of subclassOf, the most common built-in relation among concepts in OWL-DL. This causes little issue and even multiple inheritance is allowed. When a concept is subsumed by two or more other concepts, then all of the individuals of the subsumed concept are members of each parent concept. But what does it mean to be a member of a concept?

OWL-DL interprets an individual as being a member of a particular concept. Here, concepts are interpreted as sets. Set theory is a powerful modeling device used extensively in mathematics and formal concept analysis. In basic set theory, the fundamental notions of union, intersection, and subset play a key role. Furthermore, the mathematical notion of set membership should not be confused with instantiation. Gangemi et al., 2001 use the following example to illustrate the difference between set membership and instantiation. Consider two possible interpretations of "Socrates is a man":

1 Socrates belongs to the class of all human beings;

2 Socrates exhibits the property of *being a man*;

Then:

> "Usually, in mathematics, the two views are assumed to be equivalent, and a predicate is taken as coinciding with the set of entities that satisfy it. This view is however too simplistic, since in Tarskian semantics set membership is taken as a basis to decide the *truth value* of property instantiation, so the former notion is independent from the latter. The existence of a mapping between the two relations does not justify their identification: one thing is a set, another thing is a property common to the elements of a set." (Gangemi et al., 2001, p. 3)

As noted, a concept may simply be declared, be defined using other concepts and constructors, or be defined by an enumeration of its members. It is worth mentioning that the last method is not found, for example, in SUO-KIF.

Turning to individuals, these are, as expected, instantiations of concepts and can be used in ABox and mixed axioms, much in the same way as in FOL ontologies. The main difference is the presence of the TBox and ABox which places concepts and individuals in separate parts of the knowledge base. It would seem that individuals are not really a part of the ontology. In general, however, it should be noted that ontologies may include not only concepts and relations, but also individuals. Therefore, in the design of a knowledge base, its ontology will, technically speaking, contain statements of the TBox and of the ABox variety. Some description logic literature uses the term *ontology* to refer only to the TBox itself. In the modeling of linguistics, individuals such as German or Swahili are undeniably part of the ontological landscape. To sum up, even though description logics refer to the TBox as the ontology, the ontology of a given domain can and should include individuals as well.

Turning to the last DL sort, roles, it is clear that OWL-DL places the restriction on its relations in that only binary relations are allowed. Thus, it is impossible to express a fact summarized as, "feature F carries value V in feature system S," in a ternary predicate such as:

$$\mathsf{carries(F, V, S)} \tag{47}$$

Instead, one possibility would be to say that the feature has some value and that the pairing of a the feature with the value belongs to a feature system. Thus, it is the pairing itself, the predication, that is included in the feature system. However, it is not possible to predicate over relations using binary roles. A statement cannot itself be related to another entity via a role relation. This is not surprising considering that OWL-DL is a subset of FOL which itself does not permit such reification of predications. Otherwise, it would not be "first order". The closest that one could hope in a DL account of such a statement would be to include axioms summarized as "feature F has value V and that F is related to system S.

$$\mathsf{hasValue(F, V)} \tag{48}$$

$$\mathsf{inSpec(F, S)} \tag{49}$$

This issue is that pairings of features and values cannot be related within feature systems, at least not directly. Such statements about which features can bear which values are crucial, however, to many linguistic theories and should be dealt with.

Before leaving the discussion of the three basic sorts, it should be noted that roles can be grouped into hierarchies. In practical ontology building, such hierarchies are of little use and may only complicate the modeling process.

4.2 Justification for using OWL-DL

OWL is actually a family of languages: OWL-Lite, OWL-DL, and OWL-Full. OWL-DL is a compromise of expressive power and tractability. The expressive power of OWL-Lite is low, e.g. it does not include the possibility of using quantification, and is not adequate for the complexities of a knowledge-based system for linguistics. On the other hand

OWL-Full is known to be undecidable and not practical for currently available reasoning systems, in part due to the possibility of predicating over concepts, not just individuals. This means that OWL-DL is compatible with a number of inferencing engines, including the RACER (Haarslev and Möller, 2001) or Pellet (Sirin et al., 2007) reasoning system. In fact, "OWL-DL was carefully crafted to remain decidable" just for that purpose: implementation (Horrocks et al., 2003, p. 18).

In addition, due to the recent interest in description logic formalisms triggered by their intended application within the Semantic Web, there are now an increasing number of authoring and other related utilities available. One very popular ontology editor and visualization tool is *Protégé* (Noy et al., 2001, Knublauch et al., 2004).[6] Protégé is particularly useful as it has the ability to create and store ontologies in various formats, including OWL, DAML+OIL, UML, and other XML-based formats. Although not initially intended as a reasoner front-end, Protégé now comes packaged with the FaCT++ and Pellet reasoners. The most important function of these reasoners is that of consistency checking.

Finally, OWL now has a number of different serialization formats, including XML/RDF, OWL/XML, OWL Functionaly Syntax, and the Manchester OWL Syntax. These serializations are endowed with a fixed set of data types inherited from XML Schema and RDF (Smith et al., 2004). Especially in the context of creating an infrastructure for linked linguistic data Berners-Lee, 2006, XML/RDF syntax of OWL conforms to best-practice recommendations for data portability (Bird and Simons, 2003). That is, the XML/RDF version of OWL is good candidate for interoperability, at least in terms of the format, with linguistic data also in XML/RDF.

Notes

1. All references to GOLD are based on the version found at http://www.linguistics-ontology.org/.
2. Concepts that are never defined are referred to as *primitive* concepts.
3. In the table D is assumed to be a built-in data type and not a declared concept.
4. As a convention, individuals are given in small-caps and sometimes with an arbitrary number, e.g. NOUN123.
5. Recall the OWL-DL terminology for concept and role, given here in parenthesis.
6. The latest version of the Protégé tool is available for free at http://protege.stanford.edu/.

References

Baader, Franz, Calvanese, Diego, McGuinness, Debora L., Nardi, Daniele, and Patel-Schneider, Peter (2003). *The Description Logic Handbook*. Cambridge University Press.

Baader, Franz and Nutt, Werner (2003). Basic description logics. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F., editors, *The Description Logic Handbook*, chapter 2, pages 47–100. Cambridge University Press.

Berners-Lee, Tim (2006). Linked data. Published electronically at: http://www.w3.org/DesignIssues/LinkedData.html.

Bird, Steven and Simons, Gary F. (2003). Seven dimensions of portability for language documentation and description. *Language*, 79, pages 557–582.

Borgida, Alex and Brachman, Ronald J. (2003). Conceptual modeling with description logics. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F., editors, *The Description Logic Handbook*, chapter 10, pages 349–372. Cambridge University Press.

Farrar, Scott (2007). Using 'Ontolinguistics' for language description. In Schalley, Andrea and Zaefferer, Dietmar, editors, *Ontolinguistics: How ontological status shapes the linguistic coding of concepts*, pages 175–191. Mouton de Gruyter, Berlin.

Farrar, Scott and Langendoen, D. Terence (2003). A linguistic ontology for the Semantic Web. *GLOT International*, 7(3):97–100.

Gangemi, Aldo, Guarino, Nicola, Masolo, Claudio, and Oltramari, Alessandro (2001). Understanding top-level ontological distinctions. In *Proceedings of IJCAI 2001 workshop on Ontologies and Information Sharing*.

Gruber, Thomas R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Haarslev, Volker and Möller, Ralf (2001). RACER system description. In *International Joint Conference on Automated Reasoning (IJCAR'2001)*, number 2083 in Lecture Notes in Computer Science, pages 701–712, Berlin. Springer-Verlag.

Horridge, Matthew, Knublauch, Holger, Rector, Alan, Stevens, Robert, and Wroe, Chris (2004). A practical guide ot building owl ontologies using the protégé owl plugin and CO-ODE tools, edition 1.0. Technical report, The University of Manchester and Stanford University.

Horrocks, Ian, Patel-Schneider, Peter F., and van Harmelen, Frank (2003). From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1).

Horrocks, Ian, Sattler, Ulrike, and Tobies, Stephan (1999). Practical reasoning for expressive description logics. In Ganzinger, Harald, McAllester, David, and Voronkov, Andrei, editors, *Proceedings of the 6th. International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180, Berlin. Springer-Verlag.

Knublauch, Holger, Musen, Mark A., and Rector, Alan L. (2004). Editing description logic ontologies with the Protégé OWL Plugin. In *Proceedings of Description Logics 2004*.

Masolo, Claudio, Borgo, Stefano, Gangemi, Aldo, Guarino, Nicola, and Oltramari, Alessandro (2003). Ontologies library (final). WonderWeb Deliverable D18, ISTC-CNR, Padova, Italy.

Nardi, Daniele and Brachman, Ronald J. (2003). An introduction to description logics. In Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., and Patel-Schneider, P.F., editors, *The Description Logic Handbook*, chapter 1, pages 1–40. Cambridge University Press.

Niles, Ian and Pease, Adam (2001). Toward a standard upper ontology. In Welty, Chris and Smith, Barry, editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Ogunquit, Maine. Association for Computing Machinery.

Noy, Natalya F., Sintek, Michael, Decker, Stefan, Crubezy, Monica, Fergerson, Ray W., and Musen, Mark A. (2001). Creating Semantic Web contents with protege-2000. *IEEE Intelligent Systems*, 16(2):60–71.

Sirin, Evren, Parsia, Bijan, Grau, Bernardo Cuenca, Kalyanpur, Aditya, and Katz, Yarden (2007). Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53.

Smith, Michael K., Welty, Chris, and McGuinness, Deborah L. (2004). Owl web ontology language guide. W3C Recommendation 20040210, W3C. http://www.w3.org/TR/owl-guide/.