The Distributed Ontology, Modelling and Specification Language – DOL

Till Mossakowski\textsuperscript{1,2}, Oliver Kutz\textsuperscript{1}, Mihai Codescu\textsuperscript{3}, and Christoph Lange\textsuperscript{4}

\textsuperscript{1} Collaborative Research Centre on Spatial Cognition, University of Bremen
\textsuperscript{2} DFKI GmbH Bremen
\textsuperscript{3} University of Erlangen-Nürnberg
\textsuperscript{4} School of Computer Science, University of Birmingham

Abstract. There is a diversity of ontology languages in use, among them OWL, RDF, OBO, Common Logic, and F-logic. Related languages such as UML class diagrams, entity-relationship diagrams and object role modelling provide bridges from ontology modelling to applications, e.g. in software engineering and databases. Another diversity appears at the level of ontology modularity and relations among ontologies. There is ontology matching and alignment, module extraction, interpolation, ontologies linked by bridges, interpretation and refinement, and combination of ontologies. The Distributed Ontology, Modelling and Specification Language (DOL) aims at providing a unified meta language for handling this diversity. In particular, DOL provides constructs for (1) “as-is” use of ontologies formulated in a specific ontology language, (2) ontologies formalised in heterogeneous logics, (3) modular ontologies, and (4) links between ontologies. This paper sketches the design of the DOL language. DOL will be submitted as a proposal within the OntoIOp (Ontology Integration and Interoperability) standardisation activity of the Object Management Group (OMG).

1 Introduction

OWL is a popular language for ontologies.\textsuperscript{5} Yet, the restriction to a decidable description logic often hinders ontology designers from expressing knowledge that cannot (or can only in quite complicated ways) be expressed in a description logic. A practice to deal with this problem is to intersperse OWL ontologies with first-order axioms, e.g. in the case of bio-ontologies where mereological relations such as parthood are of great importance, though only partly definable in OWL. However, these remain informal annotations to inform the human designer, rather than first-class citizens of the ontology with formal semantics and impact on reasoning. One goal of the Distributed Ontology, Modelling and Specification Language (DOL), discussed in detail in this paper, is therefore to equip such heterogeneous ontologies with a precise semantics and proof theory.

\textsuperscript{5} We adopt the completely formal position that an ontology is a formal theory in a given ontology language, and that an ontology language is any logical language that some community considers suitable for ontology design.
A variety of languages is used for formalising ontologies. Some of these, such as RDF (mostly used for linked data), OBO and certain\(^6\) UML class diagrams, can be seen more or less as fragments and notational variants of OWL, while others, such as F-logic and Common Logic (CL), clearly go beyond the expressiveness of OWL.

We face this diversity not by proposing yet another ontology language that would subsume all the others, but by accepting this pluralism in ontology languages and by formulating means (on a sound and formal semantic basis) to compare and integrate ontologies written in different formalisms. This view is a bit different from that of unifying languages such as OWL and CL, which are meant to be “universal” formalisms (for a certain domain/application field), into which everything else can be mapped and represented. While such “universal” formalisms are clearly important and helpful for reducing the diversity of formalisms, it is still a matter of fact that no single formalism will be the Esperanto that is used by everybody [23]. It is therefore important to both accept the existing diversity of formalisms and to provide means of organising their coexistence in a way that enables formal interoperability among ontologies.

DOL enjoys the following distinctive features:

- modular and distributed ontologies are specially supported,
- ontologies can not only be aligned (as in BioPortal [37] and NeON [14]), but also combined along alignments,
- logical links between ontologies (interpretation of theories, conservative extensions etc.) are supported,
- support for a variety of ontology languages (OWL, RDF, Common Logic, first-order logic; planned: UML, relational database schemas, F-logic, distributed description logics, and more),
- ontologies can be translated to other ontology languages, and compared with ontologies in other languages,
- heterogeneous ontologies involving several languages can be built,
- ontology languages and ontology language translations are first-class citizens and are available on the Web as linked data.

The paper is organised as follows: we first discuss the theoretical foundations of DOL in Section 2, followed by a sketch of the DOL language itself in Section 3. Section 4 briefly discusses the DOL-enabled, web-based ontology repository engine Ontohub, and Section 5 concludes.

2 Foundations of the Distributed Ontology, Modelling and Specification Language (DOL)

The Distributed Ontology, Modelling and Specification Language (DOL)\(^7\) aims at providing a unified framework for (1) “as-is” use of ontologies formulated in

---

\(^6\) Those avoiding qualified associations (amounting to identification constraints), \(n\)-ary relations (for \(n > 2\)) and stereotyping.

\(^7\) DOL has formerly been standardised within ISO/TC 37/SC 3. The OntoIOp (Ontology Integration and Interoperability) activity is now being continued at OMG, see the project page at http://ontoiop.org.
a specific ontology language, (2) ontologies formalised in heterogeneous logics, (3) modular ontologies, and (4) links between ontologies. Historically, the design of DOL has inherited many ideas and features (1) discussed in the Workshop on Modular Ontologies series [13, 12, 39, 19, 24, 40], (2) from the Alignment API [9], and (3) from the CASL (Common Algebraic Specification Language) and HetCASL (CASL’s heterogeneous extension) languages, standardised in IFIP WG 1.3\(^8\) (Foundations of System Specification) [2, 27, 32, 20].

A distributed ontology in DOL consists of modules formalised in basic ontology languages, such as OWL (based on description logic) or Common Logic (based on first-order logic with some second-order features). These modules are serialised in the existing syntaxes of these languages in order to facilitate reuse of existing ontologies. DOL adds a meta-level on top, which allows for expressing heterogeneous ontologies and links between ontologies.\(^9\) Such links include (heterogeneous) imports and alignments, conservative extensions (important for studying ontology modules), and theory interpretations (important for reusing proofs). Thus, DOL gives ontology interoperability a formal grounding and makes heterogeneous ontologies and services based on them amenable to automated verification. The basic syntax and semantics of DOL has been introduced in [35, 34], and the general theory of heterogeneous specifications for ontologies in [22]. DOL uses internationalised resource identifiers (IRIs, the Unicode-aware superset of URIs) for all entities of distributed ontologies to make them referenceable on the Web.

2.1 Foundations

The large variety of logical languages in use can be captured at an abstract level using the concept of institutions [10]. This allows us to develop results independently of the particularities of a logical system and to use the notions of institution and logical language interchangeably throughout the rest of the paper. The main idea is to collect the non-logical symbols of the language in signatures and to assign to each signature the set of sentences that can be formed with its symbols. For each signature, we provide means for extracting the symbols it consists of, together with their kind. Signature morphisms are mappings between signatures. We do not assume any details except that signature morphisms can be composed and that there are identity morphisms; this amounts to a category of signatures. Readers unfamiliar with category theory may replace this with a partial order (signature morphisms are then just inclusions). See [34] for details of this simplified foundation.

Institutions also provide a model theory, which introduces semantics for the language and gives a satisfaction relation between the models and the sentences of a signature. The only restriction imposed is the satisfaction condition, which captures the idea that truth is invariant under change of notation (and enlargement of context) along signature morphisms. This relies on two further compo-

\(^8\) See http://ifipwg13.informatik.uni-bremen.de

\(^9\) The languages that we call “basic” ontology languages here are usually limited to one logic and do not provide meta-theoretical constructs.
nents of institutions: the translation of sentences along signature morphisms, and the reduction of models against signature morphisms (generalising the notion of model reduct known from logic).

It is also possible to complement an institution with a proof theory, introducing a derivability relation between sentences, formalised as an entailment system [30]. In particular, this can be done for all logics that have so far been in use in DOL.

Example 1. OWL signatures consist of sets of atomic classes, individuals and properties. OWL signature morphisms map classes to classes, individuals to individuals, and properties to properties. For an OWL signature $\Sigma$, sentences are subsumption relations between classes or properties, membership assertions of individuals in classes and pairs of individuals in properties, complex role inclusions, and some more. Sentence translation along a signature morphism simply replaces non-logical symbols with their image along the morphism. The kinds of symbols are class, individual, object property and data property, respectively, and the set of symbols of a signature is the union of its sets of classes, individuals and properties. Models are (unsorted) first-order structures that interpret concepts as unary and properties as binary predicates, and individuals as elements of the universe of the structure, and satisfaction is the standard satisfaction of description logics. This gives us an institution for OWL.

In this framework, a basic ontology $O$ over an institution $I$ is a pair $(\Sigma, E)$ where $\Sigma$ is a signature and $E$ is a set of $\Sigma$-sentences. Given a basic ontology $O$, we denote by $\text{Sig}(O)$ the signature of the ontology. An ontology morphism $\sigma : (\Sigma_1, E_1) \rightarrow (\Sigma_2, E_2)$ is a signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$ such that $\sigma(E_1)$ is a logical consequence of $E_2$.

Several notions of translations between institutions can be introduced. The most frequently used variant are institution comorphisms [11]. A comorphism from institution $L_1$ to institution $L_2$ maps $L_1$-signatures to $L_2$-signatures along a functor $\Phi$ and $\Sigma$-sentences in $L_1$ to $\Phi(\Sigma)$-sentences in $L_2$, for each $L_1$-signature $\Sigma$, while $\Phi(\Sigma)$-models are mapped to $\Sigma$-models. Again, a satisfaction condition has to be fulfilled. For institution morphisms, the directions of the translation of sentences and models are reversed. See [11] for full details.

Figure 1 shows a conceptual hierarchy of mappings. Mappings are split along the following dichotomies:

- translation versus projection: a translation embeds or encodes a logic into another one, while a projection is a forgetful operation (e.g. the projection from first-order logic to propositional logic forgets predicates with arity greater than zero). Technically, the distinction is that between institution comorphisms and morphisms.
- plain mapping versus simple theoroidal mapping [11]: while a plain mapping needs to map signatures to signatures, a simple theoroidal mapping maps signatures to theories. The latter therefore allows for using “infrastructure

\[\text{This graph, computed within PROTÉGÉ, shows the inferred class hierarchy below the class Mapping of the LoLa ontology (see Section 2.3 below).}\]
axioms**: e.g. when mapping OWL to Common Logic, it is convenient to rely on a first-order axiomatisation of a transitivity predicate for properties.

Mappings can also be classified according to their accuracy; see [33] for details. **Sublogics** are the most accurate mappings: they are syntactic subsets. **Embeddings** come close to sublogics, like injective functions come close to subsets. A mapping can be **faithful** in the sense that logical consequence (or logical deduction) is preserved and reflected, that is, inference systems and reasoning engines for the target logic can be reused for the source logic (along the mapping). **(Weak) exactness** is a technical property that guarantees this faithfulness even in the presence of ontology structuring operations [5].

### 2.2 A Graph of Logic Translations

Figure 2 is a revised and extended version of the graph of logics and translations introduced in [33]. New nodes include UML class diagrams, **OWL-Full** (i.e. OWL with an RDF semantics instead of description logic semantics), and Common Logic without second-order features (CL¹). We have defined the translations between most of these logics in earlier publications [35, 33]. The definitions of the DOL conformance of some central standard ontology languages and translations among them will be given as annexes to the standard and published in an open registry, which is also the place where the remaining definitions will be maintained (cf. Section 2.3).

![Fig. 1. The mapping subset of the LoLa ontology](image)
2.3 A Registry for Ontology Languages and Mappings

Beyond those shown so far, it will be possible to use any (future) language or mapping (in the sense of Section 2.1) with DOL. We host a registry to which the community can contribute descriptions of any languages and mappings\textsuperscript{11}, as well as logics and serialisations (i.e. concrete syntaxes of languages).\textsuperscript{12} The LoLa (“logics and languages”) ontology formalises these notions [25]. LoLa and its main instance, the registry, form themselves a distributed ontology. The registry is written in RDF, LoLa in OWL plus some Common Logic axioms.

Figure 3 shows the top-level classes of LoLa’s OWL module, axiomatising logics, languages, and mappings. Object-level classes (that is, classes providing the vocabulary for expressing distributed ontologies) comprise ontologies, their constituents (namely symbols and sentences), as well as links between ontologies. Mappings are modelled as shown in Figure 1: by a hierarchy of properties corresponding to the different types of edges in Figure 2. The full LoLa ontology is available at \url{http://purl.net/dol/1.0/rdf#}.

\textsuperscript{11} As distributed ontologies refer to languages and mappings by IRIs, third parties may also set up their own, decentral registry extensions.

\textsuperscript{12} The OWL 2 DL language is, e.g., exactly as expressive as the logic $SROIQ(D)$ [17], and it can be serialised in the text-based Manchester syntax or as XML.
3 The Language DOL

3.1 Motivation

Many (domain) ontologies are written in DLs such as $SROIQ$ and its profiles. These logics are characterised by having a rather fine-tuned expressiveness, exhibiting (still) decidable satisfiability problems, whilst being amenable to highly optimised implementations.

However, expressiveness beyond standard DLs is required for many foundational ontologies (as well as bio-medical ontologies), for instance DOLCE\(^{13}\), BFO\(^{14}\), or GFO\(^{15}\). Moreover, for practical purposes, these foundational ontologies also come in different versions ranging in expressiveness, typically between OWL (e.g. DOLCE Light, BFO-OWL) and first-order (DOLCE, GFO) or even second-order logic (BFO-Isabelle).

The relation between such different versions, OWL and first-order, may be recorded in various ways. In some cases it is primarily discussed in the research literature, see Keet’s mereo-topological ontology\(^ {18}\) for an example, or it is described in the OWL ontology within a comment, not carrying formal semantics. Such a comment might only contain an informal explanation of how the OWL approximation was obtained (DOLCE Light is an example), but it might also describe a fully formal, axiomatised first-order extension of the OWL ontology.

Consider the BFO-OWL object property $temporalPartOf$. The OWL axiomatisation states this to be a transitive subproperty of $occurrentPartOf$, and the inverse of $hasTemporalPart$.\(^ {16}\) This property is, however, annotated in a rich way, containing example usages, a richer first-order axiomatisation of this property with pointers to the corresponding axioms in the first-order version, as well as natural language renderings of these axioms. The logical part of this annotation may be captured in DOL as follows: an OWL ontology first lists the entire OWL axiomatisation of BFO. In a second step, we import this OWL ontology along a translation to Common Logic, and subsequently extend the resulting first-order version of BFO-OWL with the first-order axioms previously only listed as comments. We obtain a two-level specification of BFO: the original OWL part

\(^{13}\)See http://www.loa.istc.cnr.it/DOLCE.html

\(^{14}\)See http://www.ifomis.org/bfo/

\(^{15}\)See http://www.onto-med.de/ontologies/gfo/

\(^{16}\)Parthood, typically understood as an anti-symmetric relation in mereology, is the canonical example of a relation that cannot be adequately formalised in OWL; a corresponding comment can be found in many bio-medical ontologies.
(supported by OWL reasoners) and the full first-order part in Common Logic (amenable to first-order theorem proving and non-conservatively extending the OWL consequences).

3.2 DOL Syntax and Semantics

The DOL language is not “yet another ontology language”, but a meta language for expressing relations between ontologies. Therefore, any ontology written in any conforming ontology language also is a DOL ontology. This has the clear advantage that users can leave their ontologies as they are when working with DOL.

DOL provides two main abstract syntax categories:

1. Modular and heterogeneous ontologies. Such an ontology is written in a modular way, with the help of structuring operations. The semantics of ontologies is given by a signature and a class of models. In some cases, we can additionally provide a theory-level semantics of ontologies, as a signature and a class of sentences that, if it exists, agrees with the model-level semantics (that is, the model class is equal to the class of models satisfying the theory). We call an ontology flattenable if it has a theory-level semantics and elusive if it only admits a model-level semantics. This can be decided according to the outermost structuring operation on ontologies, as follows:

   **Flattenable ontologies:** basic ontologies, extension, union, translation, interpolate/forget, extract, reference, qualification, combination, bridge.
   Among these operations, interpolate/forget and extract can only be applied to flattenable ontologies.

   **Elusive ontologies:** reduction, minimisation and maximisation.

   For detailed definitions of these types of ontologies, see Section 3.3.

2. Distributed ontologies. These consist of a list of declarations involving (possibly modular and/or heterogeneous) ontologies. These declarations can be ontology definitions (assigning a name to an ontology), interpretations (specifying a logical consequence relationship between ontologies), equivalences of ontologies (specifying that their model classes are in bijective correspondence), module relations (between ontologies and their modules), ontology alignments, and qualifications of the language, logic and/or serialisation. This will be detailed in Section 3.4.

3.3 Modular and Heterogeneous Ontologies

A (possibly modular and/or heterogeneous) ontology can be one of the following:

(a) a basic ontology $O$ written inline, in a conforming ontology language and serialisation. The semantics is inherited from the ontology language. $O$ can also be an ontology fragment, which means that some of the symbols or axioms may refer to symbols declared outside $O$ (i.e. in an imported ontology). This is mainly used for extensions and equivalences. Here are two sample ontologies in OWL (using Manchester syntax) and Common Logic (using CLIF):
Class: Woman EquivalentTo: Person and Female

ObjectProperty: hasParent

(cl-module PreOrder
(forall (x) (le x x))
(forall (x y z) (if (and (le x y) (le y z)) (le x z))))

(b) an ontology qualified with the ontology language that is used to express it (written language \( l : O \), where \( l \) identifies a language). Similarly, qualifications can also be by logic (written logic \( l : O \)), and/or serialisation (written syntax \( s : O \)).

(c) an IRI reference to an ontology existing on the Web\(^\text{18}\), possibly abbreviated using prefixes.\(^\text{19}\) For example:

```
%prefix(
  co-ode: <http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/> )%
http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/pizza.owl
co-ode:pizza.owl
```

(d) an extension of an ontology by new symbols and axioms, written \( O_1 \) then \( O_2 \), where \( O_2 \) is an ontology (fragment) in a conforming ontology language. The resulting signature is that of \( O_1 \), augmented with the symbols in \( O_2 \). An extension can optionally be marked as conservative (%mcons or %ccons after the “then”). The semantics is that each \( O_1 \)-model must have at least one expansion to the whole extension \( O_1 \) then \( O_2 \) (for %mcons) resp. that each logical consequence of \( O_1 \) then \( O_2 \) is already one of \( O_1 \) if it is over the signature of \( O_1 \) (for %ccons). In case that \( O_2 \) does not introduce any new symbols, the keyword %implied can be used instead of %ccons or %mcons; the extension then merely states intended logical consequences. The keyword %def stands for definitional extensions. This is similar to %ccons, but the model expansion must always exist uniquely. The following OWL ontology is an example for the latter:

```
Class Person
Class Female
then %def

Class: Woman EquivalentTo: Person and Female
```

(e) a union of two self-contained ontologies (not fragments), written \( O_1 \) and \( O_2 \). Models of this union are those models that are (perhaps after appropriate reduction) models of both \( O_1 \) and \( O_2 \). For example, the class of commutative monoids can be expressed as

---

17 Some of the following listings omit obvious qualifications for readability.

18 Note that not all ontologies can be downloaded by dereferencing their IRIs. Implementing a catalogue mechanism in DOL-aware applications might remedy this problem.

19 Some of the following listings abbreviate IRIs using prefixes but omit the prefix bindings for readability.
Forming a union of ontologies is a particularly common operation in the RDF logic, where it is known as merging graphs [15, section 0.3]; however, the RDF language provides no explicit syntax for this operation. When multiple RDF ontologies (“graphs”) contain statements about the same symbol (“resource”), i.e., syntactically, triples having the same subject, the effect is that in the merged graph the resource will have all properties that have previously been stated about it separately. Different kinds of properties, e.g., multilingual labels, geodata, or outgoing links to external graphs, are often maintained in different RDF graphs, which are then merged; consider the following excerpt:

{ :UniBremen rdfs:label "Université de Brême"@fr . } and
{ :UniBremen geo:lat "53.108612"^^xsd:float . } and
{ :UniBremen owl:sameAs20
  <http://dbpedia.org/page/University_of_Bremen> . }

(f) a translation of an ontology to a different signature (written $O$ with $\sigma$, where $\sigma$ is a signature morphism) or into some ontology language (written $O$ with translation $\rho$, where $\rho$ is an institution comorphism). For example, we can combine an OWL ontology with a first-order axiom (formulated in Common Logic) as follows:

ObjectProperty: isProperPartOf
Characteristics: Asymmetric
SubPropertyOf: isPartOf
with translation trans:SROIQtoCL
then
  (if (and (isProperPartOf x y) (isProperPartOf y z)) (isProperPartOf x z))

Note that OWL can express transitivity, but not together with asymmetry.

(g) a reduction of an ontology to a smaller signature $\Sigma$ is written $O$ reveal $\Sigma$. Alternatively, it can be written $O$ hide $\Sigma$, where $\Sigma$ is the set of symbols to be hidden (i.e. this is equivalent to $O$ reveal $\text{Sig}(O) \setminus \Sigma$). The effect is an existential quantification over all hidden symbols. For example, when specifying a group in sorted first-order logic, using the CASL language,

sort Elem
ops 0: Elem; __+__: Elem * Elem -> Elem; inv: Elem -> Elem
forall x,y,z . 0 + x = x
. x + (y + z) = (x + y) + z
. x + inv(x) = 0
reveal Elem, 0, __+__

revealing everything except the inverse operation inv results in a specification of the class of all monoids that can be extended with an inverse operation, i.e. the class of all groups with inverse left implicit.

Here is an example of hiding:

20 While owl:sameAs is borrowed from the vocabulary of OWL, it is commonly used in the RDF logic to link to resources in external graphs, which should be treated as if their IRI were the same as the subject’s IRI.
ontology Pizza = "a simplified remake of the Pizza ontology" [16]

Individual: TomatoTopping
Individual: MozzarellaTopping DifferentFrom: TomatoTopping
ObjectProperty: hasTopping
Class: VegetarianTopping
  EquivalentTo: { TomatoTopping, MozzarellaTopping, ... }
Class: VegetarianPizza SubClassOf: some hasTopping VegetarianTopping

end

ontology Pizza_hide_VegetarianTopping =
  Pizza hide VegetarianTopping
end

A reduction to a less expressive logic is written $O$ hide along $\mu$, where $\mu$ is an institution morphism. This is a common operation in TBox/ABox settings, where an ontology in an expressive language provides the terminology (TBox) used in assertions (ABox) stated in a logic that is less expressive but scales to larger data sets: OWL DL (whose logic is SROIQ) vs. RDF is a typical language combination:

ontology TBoxABox =
  Pizza hide along trans:SROIQtoRDF
  then language lang:RDF syntax ser:RDF/Turtle : {
    :myPizza :hasTopping
    [ a :TomatoTopping ], [ a :MozzarellaTopping ] .
  }

(h) an interpolation of an ontology, either in a subsignature or a sublogic, optionally with respect to a logic $L$ (written $O$ keep in $\Sigma$ with $L$), where $\Sigma$ is a signature or a logic and $L$ is a logic)\(^21\). The effect is that sentences not expressible in $\Sigma$ are weakened or removed, but the resulting theory still has the same $L$-consequences. The “with $L$” is optional, it defaults to the logic of $O$. Technically, this is a uniform interpolant [41, 29]. In case that $\Sigma$ is a sublogic, this is also called approximation [28]. For example, we can interpolate the first-order DOLCE mereology in OWL:\(^22\)

DOLCE.Mereology keep in log:OWL

Dually, $O$ forget $\Sigma$ with $L$ interpolates $O$ with the signature $\text{Sig}(O) \setminus \Sigma$, i.e. $\Sigma$ specifies the symbols that need to be left out. Cf. the notion of forgetting in [41, 29]. For example,

Pizza forget VegetarianTopping

This has a theory-level semantics, i.e. yields a theory in the reduced signature (without VegetarianTopping). By contrast Pizza hide VegetarianTopping has a model-level semantics.

\(^{21}\) It is also possible to specify a signature and a logic simultaneously: $O$ keep in $\Sigma$, $L_1$ with $L_2$

\(^{22}\) Interpolants need not always exist, and even if they do, tools might only be able to approximate them.
(i) a module extracted from an ontology, written \( O \extract c \Sigma \with m \). Here, \( \Sigma \) is a restriction signature (which needs to be a subsignature of \( \Sig(O) \)), \( c \) is one of \%mcons and \%ccons, and \( m \) identifies a module extraction method.

The extracted module is a subontology of \( O \) with signature larger than (or equal to) \( \Sigma \), such that \( O \) is a conservative extension of the extracted module.

Dually, \( O \remove c \Sigma \with m \) extracts w.r.t. the signature \( \Sig(O) \setminus \Sigma \).

For example, using the syntactic locality-* extraction method [38]:

\[
\text{Pizza remove } \%mcons \\
\text{VegetarianTopping} \\
\text{with } \langle \text{http://example.org/onto/module/syntactic-locality-*} \rangle
\]

Table 1 illustrates some of the connections between (g)-(i). We have three ways of removing the class VegetarianTopping from the ontology Pizza: (1) using hiding, we keep the model class of Pizza, but just remove the interpretation of VegetarianTopping from each model. Note that the resulting ontology has

\[
\text{VegetarianPizza SubClassOf:} \\
\text{Annotations: dol:iri (*)} \\
\text{some hasTopping \{ TomatoTopping, MozzarellaTopping, ... \}}
\]

as a logical consequence. This is also a consequence of the corresponding uniform interpolant

\[
\text{Pizza forget VegetarianTopping}
\]

which captures the theory of Pizza hide VegetarianTopping. Note that there is a subtle difference between (model-theoretic) hiding and (consequence-theoretic) forgetting: a model satisfying the theory of \( O \) hide \( \Sigma \) might itself not be a model of \( O \) hide \( \Sigma \). In examples involving “\( \with L \)”, the uniform interpolant can be weaker than the hiding, because it is only required to have the same logical consequences in some language \( L \), and a formula like (\*) might not be a formula of \( L \). Finally, an extracted module does not contain (\*), because it only selects a subontology, and Pizza does not contain (\*).

Note that while forget/keep and hide/reveal both work w.r.t. smaller signatures and sublogics, remove/extract does not work for sublogics. This is because remove/extract must always respect the conservative extension property, which may not be possible when projecting to a sublogic. And if conservativity cannot be guaranteed, then forget/keep can be used in any case.

(j) a combination of ontologies, written combine \( O_1, \ldots, O_n, L_1, \ldots, L_m \). Here the \( L_j \) are links between ontologies, see below. For disambiguating the symbols in the combined ontology, the individual ontologies can be prefixed with labels, like \( n : O \), which are scoped to the current distributed ontology. The simplest example of a combination is a disjoint union (we here translate OWL ontologies into many-sorted OWL in order to be able to distinguish between different universes of individuals):

\[\text{Note that the resulting module can still contain symbols from } \Sigma, \text{ because the resulting signature may be enlarged.}\]
<table>
<thead>
<tr>
<th></th>
<th>remove/extract</th>
<th>forget/keep</th>
<th>hide/reveal</th>
</tr>
</thead>
<tbody>
<tr>
<td>semantic background</td>
<td>conservative</td>
<td>uniform</td>
<td>model reduct</td>
</tr>
<tr>
<td>extension</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relation to original</td>
<td>subtheory</td>
<td>interpretable</td>
<td>interpretable</td>
</tr>
<tr>
<td>approach</td>
<td>theory level</td>
<td>theory level</td>
<td>model level</td>
</tr>
<tr>
<td>type of ontology</td>
<td>flattenable</td>
<td>flattenable</td>
<td>elusive</td>
</tr>
<tr>
<td>signature of result</td>
<td>$\geq \Sigma$</td>
<td>$\Sigma$</td>
<td>$\Sigma$</td>
</tr>
<tr>
<td>change of logic</td>
<td>not possible</td>
<td>possible</td>
<td>possible</td>
</tr>
</tbody>
</table>

| Table 1. Extract – Forget – Hide |

ontology Publications1 =
  Class: Publication
  Class: Article SubClassOf: Publication
  Class: InBook SubClassOf: Publication
  Class: Thesis SubClassOf: Publication
  ...

ontology Publications2 =
  Class: Thing
  Class: Article SubClassOf: Thing
  Class: BookArticle SubClassOf: Thing
  Class: Publication SubClassOf: Thing
  Class: Thesis SubClassOf: Thing
  ...

ontology Publications_Combined =
  combine
  1 : Publications1 with translation trans:OWL2MS-OWL,
  2 : Publications2 with translation trans:OWL2MS-OWL
  %% implicitly: Article $\mapsto$ 1:Article ...
  %
  %
  %
  Article $\mapsto$ 2:Article ...
end

(This example will be continued using bridges below.) If links or alignments are present, the semantics of a combination is a quotient of a disjoint union (aligned symbols are identified). Technically, this is a colimit, see [42, 6]. An example for this is given along with the examples for alignments below.

(k) a minimisation of an ontology imposes a closed-world assumption on part of the ontology. It forces the non-logical symbols declared in $O$ to be interpreted in a minimal way. This is written minimize $\{ O \}$. Symbols declared before the minimised part are considered to be fixed for the minimisation (that is, we minimise among all models with the same reduct). Symbols declared after the minimisation can be varied. This is borrowed from circumscription [26, 3]. Alternatively, the non-logical symbols to be minimised and to be varied can be explicitly declared: $O$ minimize $\Sigma_1$ vars $\Sigma_2$. For example, in
the following OWL theory, B2 is a block that is not abnormal, because it is not specified to be abnormal, and hence it is also on the table.

```
Class: Block
Individual: B1 Types: Block
Individual: B2 Types: Block DifferentFrom: B1
then minimize {
   Class: Abnormal
   Individual: B1 Types: Abnormal }
then
Class: OnTable
Class: BlockNotAbnormal EquivalentTo:
   Block and not Abnormal SubClassOf: OnTable
then %implied
Individual: B2 Types: OnTable
```

Dually to minimisations, there are also maximisations.

(l) an ontology bridge, written \(O_1\) bridge with translation \(t\) \(O_2\), where \(t\) is a logic translation. The semantics is that of \(O_1\) with translation \(t\) then \(O_2\). Typically, \(t\) will translate a language like OWL to some language for distributed description logic or \(\mathcal{E}\)-connections \([4, 21, 8]\), and \(O_2\) introduces some axioms involving the relations (introduced by \(t\)) between ontologies in \(O_1\). For example,

```
Publications_Combined
bridge with translation trans:MS-OWL2DDL
   % implicitly added by translation trans:MS-OWL2DDL:
   % binary relation providing the bridge
1:Publication ⊑\(\rightarrow\) 2:Publication
1:PhdThesis ⊑\(\rightarrow\) 2:Thesis
1:InBook ⊑\(\rightarrow\) 2:BookArticle
1:Article ⊑\(\rightarrow\) 2:Article
1:Article ⊑\(\rightarrow\) 2:Article
```

### 3.4 Distributed Ontologies

Distributed ontologies. These have an optional identifier, declared with distributed ontology \(Id\), and consist of

(a) ontology definitions, written ontology \(Id = O\). For example,

```
ontology co-code:Pizza =
   Class: VegetarianPizza
   Class: VegetableTopping
   ObjectProperty: hasTopping
...
end
```
(b) theory interpretations, written interpretation Id : $O_1 \text{ to } O_2 = \sigma$, expressing that the $\sigma$-reduct of each model of $O_2$ is a model of $O_1$. Instead of $\sigma$, an institution comorphism can be referred to. For example, we can express that the natural numbers are a total order as follows:

```
interpretation i : TotalOrder to Nat = Elem \mapsto Nat
```

Here is a more complex example in Common Logic from the COLORE repository [7]:

```
interpretation geometry_of_time %mcons :
  %% Interpretation of linearly ordered time intervals...
  int:owltime_le
  %% ... that begin and end with an instant as lines
  %% that are incident with linearly ...
  to { ord:linear_ordering and bi:complete_graphical
    %% ... ordered points in a special geometry, ...
    and int:mappings/owltime_interval_reduction }
  = int:ProperInterval \mapsto int:Interval
```

c) ontology equivalences, written equivalence Id : $O_1 \leftrightarrow O_2 = O_3$ along $\rho_1, \rho_2$, expressing that $O_1$ and $O_2$ have model classes that are in bijective correspondence. This is done by providing a (fragment) ontology $O_3$ such that $\rho_i(O_i)$ then $O_3$ is a definitional extension [22]. $\rho_1$ and $\rho_2$ are optional institution comorphisms that default to the identity. For example, Boolean algebras are equivalent to Boolean rings:

```
equivalence e : algebra:BooleanAlgebra \leftrightarrow algebra:BooleanRing =
  \forall x,y .
  . x \land y = x*y
  . x \lor y = x + y + x*y
  . \neg x = 1 + x
  . x*y = x \land y,
  . x+y = (x \lor y) \land \neg(x \land y).
```

d) module relations, written module Id $c : O_1 \text{ of } O_2$ for $\Sigma$. This expresses that $O_1$ is a module of $O_2$ with restriction signature $\Sigma$ and conservativity $c$. If $c$ is %mcons, this means that every $\Sigma$-reduct of an $O_1$-model can be expanded to an $O_2$-model. If $c$ is %cons, this means that every $\Sigma$-sentence $\varphi$ following from $O_2$ already follows from $O_1$. This relation shall hold for any module $O_1$ extracted from $O_2$ using the extract construct.

(e) alignment definitions, written alignment Id $\text{card}_1$ $\text{card}_2 : O_1 \text{ to } O_2 = c_1, \ldots, c_n$, where $\text{card}_1$ resp. $\text{card}_2$ specify constraints on the alignment relation concerning the source resp. target. Each $\text{card}_i$ is one of $1$, $\?$, $+$, $*$ ($'1'$ for injective and total, $'+'$ for total, $'\?$' for injective and $'*'\text{ for none}$). The $c_j$ are correspondences of form $\text{sym}_1 \text{ rel } conf \text{ sym}_2$. Here, $\text{sym}_i$ is a symbol from $O_i$, $\text{rel}$ is one of the built-in relations $>$, $<$, $=\text{, } \exists\text{, } \in\text{, } \mapsto$, or an identifier of a relation specified externally, and $\text{conf}$ is an (optional) confidence value between 0 and 1. This syntax of alignments follows the
Alignment API [9]. Alignments have no formal semantics, but they can be used in combinations. For example,

```xml
.prefix() : <http://www.example.org/alignment#>
lang: <http://purl.net/dol/languages/>
ser: <http://purl.net/dol/serializations/>
trans: <http://purl.net/dol/translations/> %

distributed ontology Alignments

language lang:OWL2/DL syntax ser:OWL2/Manchester

alignment Alignment1 : { Class: Woman } to { Class: Person } =
    Woman < Person
end

ontology AlignedOntology1 =
    combine Alignment1
end

ontology Onto1 =
    Class: Person
    Class: Woman SubClassOf: Person
    Class: Bank
end

ontology Onto2 =
    Class: HumanBeing
    Class: Woman SubClassOf: HumanBeing
    Class: Bank
end

alignment VAlignment : Onto1 to Onto2 =
    Person = HumanBeing,
    Woman = Woman
end

ontology VAlignedOntology =
    combine 1 : Onto1, 2 : Onto2, VAlignment
    % 1:Person is identified with 2:HumanBeing
    % 1:Woman is identified with 2:Woman
    % 1:Bank and 2:Bank are kept distinct
end

ontology VAlignedOntologyRenamed =
    VAlignedOntology with 1:Bank \mapsto RiverBank, 2:Bank \mapsto FinancialBank,
end
```

Note that BioPortal’s [37] mappings are correspondences in the sense of the Alignment API and hence of DOL. BioPortal only allows users to collect correspondences, but not to group them into alignments. In a sense, for each pair of ontologies, all BioPortal users contribute to a big alignment between these.
(f) qualifications choosing the ontology language, logic, and/or serialisation.

This is written language_id, logic_id and/or syntax_id, referring to entries of a registry as explained in Section 2.3, and affects the subsequent definitions in the distributed ontology.

This completes our overview of DOL. The full syntax and semantics of DOL will be available at wiki.ontohub.org and later submitted to OMG for standardisation.

Note that we have not covered the role of annotations in DOL so far. For structured annotation of ontologies and their parts, e.g. with metadata, or possibly with ontological relations not built into DOL’s syntax, DOL does not provide its own syntax, but relies on the existing RDF standard. DOL allows for giving identifiers to all entities of distributed ontologies and basic ontologies and thus enables their annotation. Annotations can be maintained in an RDF ontology that is a part of the distributed ontology.

4 The Ontology Repository Ontohub

Ontohub (see http://ontohub.org) is a web-based repository engine for ontologies that are written either in DOL or in some specific ontology language.26

Ontohub provides means for organising ontologies into repositories. The distributed nature enables communities to share and exchange their contributions easily. The heterogeneous nature makes it possible to integrate ontologies written in various ontology languages. Ontohub supports a wide range of DOL-conforming ontology languages building on DOL and also supports DOL’s interpretations, equivalences and alignments. Users of Ontohub can upload, browse, search and annotate single and distributed ontologies in various languages via a web front end. Figure 4 shows an excerpt of

![Fig. 4. ontohub.org: overview of logics](http://ontohub.org)

Logics
25 Logics found
1 2 Next Last>
25 3 per page

**Name**
- OWL2
- CommonLogic
- CSL
- DDL-(OWL)
- DL-Lite-R
- E-Connections-(FOLld)
- E-Connections-(OWL)

When a basic ontology language has no mechanism for annotating or assigning identifiers to some ontology entities (as with imports in OWL or sentences in Common Logic), DOL provides a special comment syntax for injecting identifiers into basic ontologies written inline. Where identifiers in a basic ontology language are not IRIs, DOL allows for making them accessible as IRIs.

Ontohub’s sources are freely available at https://github.com/ontohub/ontohub.
the 25 logics currently available in Ontohub.

The parsing and inference back end is the Heterogeneous Tool Set (Hets [31, 36], available at hets.dfki.de). Hets supports a large number of basic ontology languages and logics, as well as the DOL meta language as described in this paper. The structural information extracted from DOL ontologies by Hets is stored in the Ontohub database and exposed to human users via a web interface and to machine clients as linked data.\textsuperscript{28}

5 Conclusion and Future Work

The Distributed Ontology, Modelling and Specification Language (DOL) integrates different lines of research that have been reflected in the WoMO community (see [13, 12, 39, 19, 24, 40]):

- conservative extensions,
- ontology module extraction,
- ontology alignments,
- combinations of ontologies along alignments,
- distributed description logics,
- $\mathcal{E}$-connections, and
- relations between ontologies written in different languages (e.g. OWL and FOL).

DOL provides a unified meta language for these (and more) concepts, with a clean formal semantics. Tool support is provided by the Heterogeneous Tool Set (Hets) and by ontohub.org. The latter will also be used for the FOIS 2014 ontology competition. Since ontologies used in FOIS papers often need expressiveness beyond OWL, the multi-logic nature of DOL and Ontohub is essential.

A number of open problems and challenges remain:

- What is a suitable abstract meta framework for non-monotonic logics and rule languages such as RIF and RuleML? Are institutions suitable here? Are the modularity questions for these languages different from those for OWL?
- What is a useful abstract notion of ontology query (language)? How to handle answer substitutions in a logic-agnostic way?
- Can the notions of class hierarchy and of satisfiability of a class be generalised from OWL to other languages?
- Can logical frameworks be used for the specification of ontology languages and translations?

\textsuperscript{27} Some (but only few) of DOL’s features are still being implemented at the time of the writing of this paper.

\textsuperscript{28} “Linked data” is a set of best practises for publishing structured data on the Web in a machine-friendly way [1]. DOL and Ontohub conform with linked data.
Acknowledgements

The development of DOL is supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 “Spatial Cognition”. Mihai Codescu was supported by the DFG, project SCHR1118-7-1. Christoph Lange was supported by EPSRC grant EP/J007498/1. The authors would like to thank the OntoIOp working group for their valuable input, particularly Michael Grüninger, Maria Keet, Fabian Neuhaus and Peter Yim. We also want to thank Carsten Lutz and Thomas Schneider for valuable input on interpolation and module extraction.

References


