The Distributed Ontology, Modelling and Specification Language – DOL

Till Mossakowski[1,2], Oliver Kutz[1], Mihai Codescu[3], and Christoph Lange[4]

[1] Collaborative Research Centre on Spatial Cognition, University of Bremen
[2] DFKI GmbH Bremen
[3] University of Erlangen-Nürnberg
[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.[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.

[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 components of institutions: the translation of sentences along signature morphisms, and

[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.

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 Σ, 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 (Σ, E) where Σ is a signature and E is a set of Σ-sentences. Given a basic ontology O, we denote by $\mathsf{Sig}(O)$ the signature of the ontology. An ontology morphism $\sigma : (\Sigma_1, E_1) \to (\Sigma_2, E_2)$ is a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ 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 Φ and Σ-sentences in L_1 to $\Phi(\Sigma)$-sentences in L_2, for each L_1-signature Σ, while $\Phi(\Sigma)$-models are mapped to Σ-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.[10] 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 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.

[10] This graph, computed within PROTÉGÉ, shows the inferred class hierarchy below the class `Mapping` of the LoLa ontology (see Section 2.3 below).

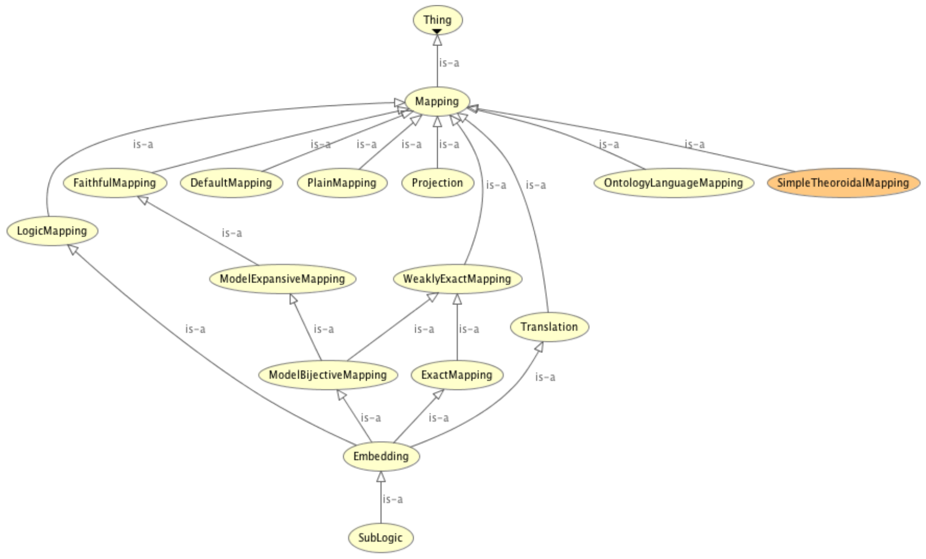

Fig. 1. The mapping subset of the LoLa ontology

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 presences 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 (CL^-). 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).

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

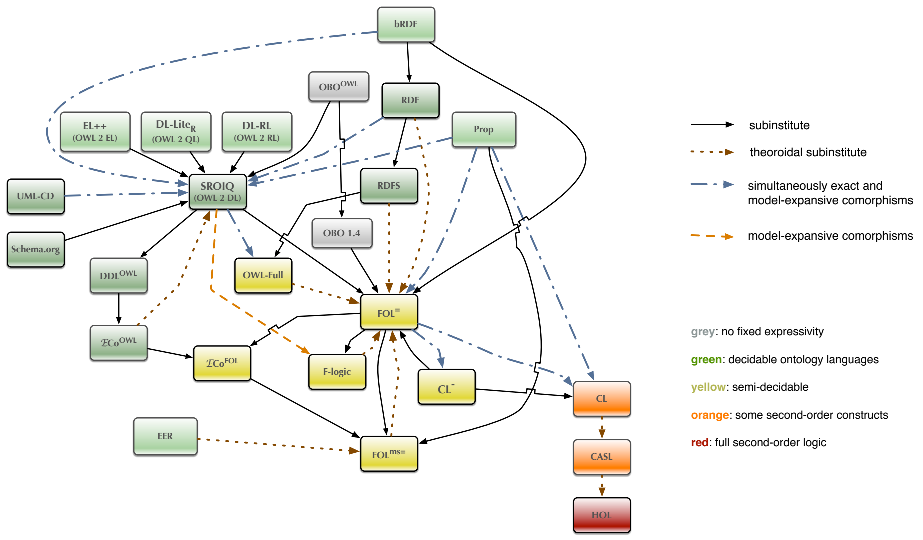

Fig. 2. The current logic translation graph for DOL-conforming languages

the community can contribute descriptions of any languages and mappings[11], as well as logics and serialisations (i.e. concrete syntaxes of languages).[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.

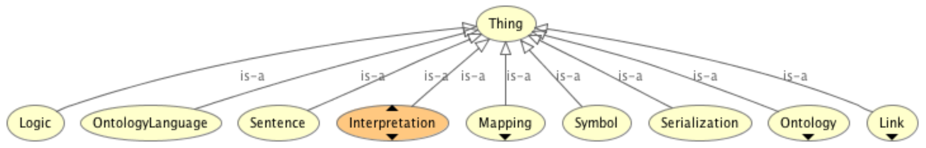

Fig. 3. Top-level classes in LoLa's OWL module

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 `http://purl.net/dol/1.0/rdf#`.

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

[12] The OWL 2 DL language is, e.g., exactly as expressive as the logic $\mathcal{SROIQ}(D)$ [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 $\mathcal{SROIQ}$ 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 (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

[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.

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 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** $s : O$).[17]

(c) an IRI reference to an ontology existing on the Web[18], possibly abbreviated using prefixes.[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. A model of an extension ontology is a model of this signature, that satisfies the axioms on O_2 and is (when appropriately reduced) a model of O_1. 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 %mcons, 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

```
algebra:Monoid and algebra:Commutative
```

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

[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.

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:sameAs[20]
    <http://dbpedia.org/page/University_of_Bremen> . }
```

(f) a *translation* of an ontology to a different signature (written O **with** σ, where σ is a signature morphism) or into some ontology language (written O **with translation** ρ, where ρ 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 Σ is written O **reveal** Σ. Alternatively, it can be written O **hide** Σ, where Σ is the set of symbols to be hidden (i.e. this is equivalent to O **reveal** $\mathsf{Sig}(O) \setminus \Sigma$). 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:

```
ontology Pizza = %% a simplified remake of the Pizza ontology [16]
  Individual: TomatoTopping
  Individual: MozzarellaTopping DifferentFrom: TomatoTopping
  ObjectProperty: hasTopping
  Class: VegetarianTopping
    EquivalentTo: { TomatoTopping, MozzarellaTopping, ... }
```

[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.

```
  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** μ, where μ 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 $\mathcal{SROIQ}$) 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** Σ **with** L, where Σ is a signature or a logic and L is a logic)[21]. The effect is that sentences not expressible in Σ 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 Σ 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** Σ **with** L interpolates O with the signature $\mathsf{Sig}(O) \setminus \Sigma$, i.e. Σ 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.

(i) a module *extracted* from an ontology, written O **extract** c Σ **with** m. Here, Σ is a restriction signature (which needs to be a subsignature of $\mathsf{Sig}(O)$), 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) Σ, such that O is a conservative extension of the extracted module.

[21] It is also possible to specify a signature and a logic simultaneously: O **keep in** Σ, L_1 **with** L_2

[22] Interpolants need not always exist, and even if they do, tools might only be able to approximate them.

Dually, O **remove** c Σ **with** m extracts w.r.t. the signature $\mathsf{Sig}(O) \setminus \Sigma$.[23] For example, using the syntactic locality-* extraction method [38]:

```
Pizza remove %mcons
  VegetarianTopping
  with <http://example.org/onto/module/syntactic-locality-*>
```

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

```
VegetarianPizza SubClassOf:
  Annotations: dol:iri (*)
  some hasTopping { TomatoTopping, MozzarellaTopping, ... }
```

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

```
  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** Σ might itself not be a model of O **hide** Σ. 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):

```
ontology Publications1 =
  Class: Publication
  Class: Article SubClassOf: Publication
  Class: InBook SubClassOf: Publication
```

[23] Note that the resulting module can still contain symbols from Σ, because the resulting signature may be enlarged.

	remove/extract	forget/keep	hide/reveal
semantic background	conservative extension	uniform interpolation	model reduct
relation to original	subtheory	interpretable	interpretable
approach	theory level	theory level	model level
type of ontology	flattenable	flattenable	elusive
signature of result	$\geq \Sigma$	$= \Sigma$	$= \Sigma$
change of logic	not possible	possible	possible

Table 1. Extract – Forget – Hide

```
  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 ↦ 1:Article ...
  %%             Article ↦ 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** Σ_1 **vars** Σ_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 $\xrightarrow{\sqsubseteq}$ 2:Publication
  1:PhdThesis $\xrightarrow{\sqsubseteq}$ 2:Thesis
  1:InBook $\xrightarrow{\sqsubseteq}$ 2:BookArticle
  1:Article $\xrightarrow{\sqsubseteq}$ 2:Article
  1:Article $\xrightarrow{\sqsupseteq}$ 2:Article
end
```

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** $Id : O_1$ **to** $O_2 = \sigma$, expressing that the σ-reduct of each model of O_2 is a model of O_1. Instead of σ, 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 ↦ 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 ↦ int:Interval end
```

(c) ontology *equivalences*, written **equivalence** *Id* : $O_1 \leftrightarrow O_2 = O_3$ **along** ρ_1, ρ_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]. ρ_1 and ρ_2 are optional institution comorphisms that default to the identity. For example, Boolean algebras are equivalent to Boolean rings:

```
equivalence e : algebra:BooleanAlgebra ↔ algebra:BooleanRing =
∀ x,y
. x ∧ y = x*y
. x ∨ y = x + y + x*y
. ¬x = 1 + x
. x*y = x ∧ y,
. x+y = (x ∨ y) ∧ ¬(x ∧ y).
end
```

(d) *module* relations, written **module** *Id c* : O_1 **of** O_2 **for** Σ. This expresses that O_1 is a module of O_2 with restriction signature Σ and conservativity c. If c is %mcons, this means that every Σ-reduct of an O_1-model can be expanded to an O_2-model. If c is %ccons, this means that every Σ-sentence φ 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* $card_1$ $card_2$: O_1 **to** O_2 = $c_1, \dots, c_n$, where $card_1$ resp. $card_2$ specify constraints on the alignment relation concerning the source resp. target. Each $card_i$ is one of 1, ?, +, * ('1' for injective and total, '+' for total, '?' for injective and '*' for none). The c_j are correspondences of form sym_1 *rel conf* sym_2. Here, sym_i is a symbol from O_i, *rel* is one of the built-in relations $>$, $<$, $=$, %, $\ni$, $\in$, $\mapsto$, or an identifier of a relation specified externally, and *conf* is an (optional) confidence value between 0 and 1. This syntax of alignments follows the Alignment API [9].[24] Alignments have no formal semantics, but they can be used in combinations. For example,

```
%prefix( :      <http://www.example.org/alignment#>
```

[24] 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.

```
         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 ↦ RiverBank, 2:Bank ↦ FinancialBank,
                       Person_HumanBeing ↦ Person
end
```

(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[25] 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 the 25 logics currently available in Ontohub.

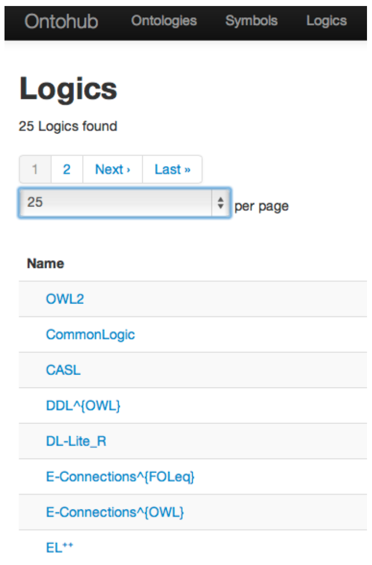

Fig. 4. `ontohub.org`: overview of logics

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

[25] 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.

[26] Ontohub's sources are freely available at `https://github.com/ontohub/ontohub`.

as described in this paper.[27] 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.[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?

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

[27] Some (but only few) of DOL's features are still being implemented at the time of the writing of this paper.

[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.

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

1. Berners-Lee, T. Design Issues: Linked Data. July 27, 2006. http://www.w3.org/DesignIssues/LinkedData.html (visited on 2010-01-20).
2. Bidoit, M. and Mosses, P. D. *CASL User Manual.* LNCS (IFIP Series) 2900. Freely available at http://www.cofi.info. Springer, 2004.
3. Bonatti, P. A., Lutz, C., and Wolter, F. The Complexity of Circumscription in DLs. In *J. Artif. Intell. Res. (JAIR) 35* (2009), pp. 717–773.
4. Borgida, A. and Serafini, L. Distributed Description Logics: Assimilating Information from Peer Sources. In *Journal of Data Semantics 1* (2003), pp. 153–184.
5. Borzyszkowski, T. Logical systems for structured specifications. In *Theoretical Computer Science 286* (2002), pp. 197–245.
6. Codescu, M. and Mossakowski, T. Heterogeneous colimits. In *MoVaH'08 Workshop on Modeling, Validation and Heterogeneity.* Ed. by F. Boulanger, C. Gaston, and P.-Y. Schobbens. IEEE press, 2008. http://www.computer.org/portal/web/csdl/abs/proceedings/icstw/2008/3388/00/3388toc.htm.
7. COLORE. An open repository of first-order ontologies represented in Common Logic. http://colore.googlecode.com.
8. Cuenca Grau, B., Parsia, B., and Sirin, E. Ontology Integration Using $\mathcal{E}$-connections. In *Modular Ontologies—Concepts, Theories and Techniques for Knowledge Modularization.* Ed. by H. Stuckenschmidt, C. Parent, and S. Spaccapietra. Vol. 5445. LNCS. Springer, 2009.
9. David, J., Euzenat, J., Scharffe, F., and dos Santos, C. T. The Alignment API 4.0. In *Semantic Web 2*, 1 (2011), pp. 3–10.
10. Goguen, J. A. and Burstall, R. M. Institutions: Abstract Model Theory for Specification and Programming. In *Journal of the Association for Computing Machinery 39* (1992). Predecessor in: LNCS 164, 221–256, 1984., pp. 95–146.
11. Goguen, J. and Roşu, G. Institution morphisms. In *Formal aspects of computing 13* (2002), pp. 274–307.
12. Grau, B. C., Honavar, V., Schlicht, A., and Wolter, F., eds. Proceedings of the 2nd International Workshop on Modular Ontologies, WoMO 2007, Whistler, Canada, October 28, 2007. Vol. 315. CEUR Workshop Proceedings. CEUR-WS.org, 2008.
13. Haase, P., Honavar, V., Kutz, O., Sure, Y., and Tamilin, A., eds. Proceedings of the 1st International Workshop on Modular Ontologies, WoMO'06, co-located with the International Semantic Web Conference, ISWC'06 November 5, 2006, Athens, Georgia, USA. Vol. 232. CEUR Workshop Proceedings. CEUR-WS.org, 2007.
14. The NeOn Ontology Engineering Toolkit. http://www.neon-project.org/. 2008. http://watson.kmi.open.ac.uk/Downloads%20and%20Publications_files/neon-toolkit.pdf.
15. Hayes, P. RDF Semantics. W3C Recommendation. World Wide Web Consortium (W3C), Feb. 10, 2004. http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

16. Horridge, M. Protégé OWL Tutorial. Version v1.3. Mar. 24, 2011. `http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/`.
17. Horrocks, I., Kutz, O., and Sattler, U. The Even More Irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)*. AAAI Press, June 2006, pp. 57–67.
18. Keet, C. M. and Artale, A. Representing and reasoning over a taxonomy of part–whole relations. In *Applied Ontology 3*, 1 (2008), pp. 91–110.
19. Kutz, O., Hois, J., Bao, J., and Cuenca Grau, B., eds. *Modular Ontologies—Proceedings of the Fourth International Workshop (WoMO 2010)*. Vol. 210. Frontiers in Artificial Intelligence and Applications. Toronto, Canada: IOS Press, 2010.
20. Kutz, O., Lücke, D., Mossakowski, T., and Normann, I. The OWL in the CASL—Designing Ontologies Across Logics. In *OWL: Experiences and Directions, 5th International Workshop (OWLED-08)*. Ed. by C. Dolbear, A. Ruttenberg, and U. Sattler. co-located with ISWC-08, Karlsruhe, Germany, October 26–27: CEUR-WS, Vol-432, 2008.
21. Kutz, O., Lutz, C., Wolter, F., and Zakharyaschev, M. $\mathcal{E}$-connections of Abstract Description Systems. In *Artificial Intelligence 156*, 1 (2004), pp. 1–73.
22. Kutz, O., Mossakowski, T., and Lücke, D. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. In *Logica Universalis 4*, 2 (2010). Special issue on 'Is Logic Universal?'
23. Kutz, O., Lange, C., Mossakowski, T., Keet, C. M., Neuhaus, F., and Grüninger, M. The Babel of the Semantic Web Tongues – In Search of the Rosetta Stone of Interoperability. In *What will the Semantic Web look like 10 Years from now? Workshop at ISWC*. Ed. by F. van Harmelen, J. A. Hendler, P. Hitzler, K. Janowicz, and D. Vrandečić. 2012. `http://stko.geog.ucsb.edu/sw2022/`.
24. Kutz, O. and Schneider, T., eds. *Modular Ontologies—Proceedings of the Fifth International Workshop (WoMO 2011)*. Vol. 230. Frontiers in Artificial Intelligence and Applications. IOS Press, 2011.
25. Lange, C., Mossakowski, T., and Kutz, O. LoLa: A Modular Ontology of Logics, Languages, and Translations. In. Ed. by T. Schneider and D. Walther. Vol. 875. CEUR-WS, 2012. `http://ceur-ws.org/Vol-875`.
26. Lifschitz, V. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*. Vol. 3. Oxford University Press, 1994, pp. 297–352.
27. Lüttich, K., Masolo, C., and Borgo, S. Development of Modular Ontologies in CASL. In *WoMO*. Ed. by P. Haase, V. Honavar, O. Kutz, Y. Sure, and A. Tamilin. Vol. 232. CEUR Workshop Proceedings. CEUR-WS.org, 2006.
28. Lutz, C., Seylan, I., and Wolter, F. An Automata-Theoretic Approach to Uniform Interpolation and Approximation in the Description Logic EL. In *KR*. Ed. by G. Brewka, T. Eiter, and S. A. McIlraith. AAAI Press, 2012.
29. Lutz, C. and Wolter, F. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In *IJCAI*. 2011, pp. 989–995.
30. Meseguer, J. General Logics. In *Logic Colloquium '87*. Ed. by H. J. Ebbinghaus. North Holland, 1989, pp. 275–329.
31. Mossakowski, T. Hets: the Heterogeneous Tool Set. `http://hets.dfki.de` (visited on 2012-12-10).
32. Mossakowski, T., Haxthausen, A., Sannella, D., and Tarlecki, A. CASL: The Common Algebraic Specification Language. In *Logics of Formal Specification Languages*. Ed. by M. H. D. Bjorner. Monographs in Theoretical Computer Science. Springer-Verlag Heidelberg, 2008. Chap. 3, pp. 241–298. `http://dx.doi.org/10.1007/978-3-540-74107-7_5`.

33. Mossakowski, T. and Kutz, O. The Onto-Logical Translation Graph. In *Modular Ontologies*. Ed. by O. Kutz and T. Schneider. IOS, 2011.
34. Mossakowski, T., Kutz, O., and Lange, C. Semantics of the distributed ontology language: Institutes and Institutions. In *Recent Trends in Algebraic Development Techniques, 21th International Workshop, WADT 2012*. Ed. by N. Martí-Oliet and M. Palomino. Vol. 7841. Lecture Notes in Computer Science. Springer, 2013, pp. 212–230. `http://link.springer.com/chapter/10.1007/978-3-642-37635-1_13`.
35. Mossakowski, T., Lange, C., and Kutz, O. Three Semantics for the Core of the Distributed Ontology Language. In *7th International Conference on Formal Ontology in Information Systems (FOIS)*. Ed. by M. Donnelly and G. Guizzardi. Vol. 239. Frontiers in Artificial Intelligence and Applications. FOIS Best Paper Award. IOS Press, 2012, pp. 337–352.
36. Mossakowski, T., Maeder, C., and Lüttich, K. The Heterogeneous Tool Set. In *TACAS 2007*. Ed. by O. Grumberg and M. Huth. Vol. 4424. Lecture Notes in Computer Science. Springer-Verlag Heidelberg, 2007, pp. 519–522.
37. Noy, N. F., Shah, N. H., Patricia L. Whetzel, ., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C. G., and Musen, M. A. BioPortal: ontologies and integrated data resources at the click of a mouse. In *Nucleic Acids Research 37* (2009). `http://bioportal.bioontology.org`, W170–W173.
38. Sattler, U., Schneider, T., and Zakharyaschev, M. Which Kind of Module Should I Extract? In *Proceedings 22nd Int. Workshop on Description Logics (DL)*. Vol. 477. CEUR Workshop Proceedings. CEUR-WS.org, 2009.
39. Sattler, U. and Tamilin, A., eds. Workshop on Ontologies: Reasoning and Modularity (WORM-08). Vol. Vol-348. (ESWC) Tenerife, Spain: CEUR Workshop Proceedings, 2008. `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-348/`.
40. Schneider, T. and Walther, D., eds. Proc. of the 6h Int. Workshop on Modular Ontologies. Vol. 875. CEUR-WS, 2012.
41. Wang, Z., Wang, K., Topor, R. W., and Pan, J. Z. Forgetting for knowledge bases in DL-Lite. In *Ann. Math. Artif. Intell. 58*, 1–2 (2010), pp. 117–151.
42. Zimmermann, A., Krötzsch, M., Euzenat, J., and Hitzler, P. Formalizing Ontology Alignment and its Operations with Category Theory. In *Proc. of FOIS-06*. 2006, pp. 277–288.