LoLa: A Modular Ontology of Logics, Languages, and Translations

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

[1] School of Computer Science, University of Birmingham
[2] Research Centre on Spatial Cognition, University of Bremen
[3] DFKI GmbH Bremen

Abstract. The Distributed Ontology Language (DOL), currently being standardised within the OntoIOp (Ontology Integration and Interoperability) activity of ISO/TC 37/SC 3, aims at providing a unified framework for (1) ontologies formalised in heterogeneous logics, (2) modular ontologies, (3) links between ontologies, and (4) annotation of ontologies. This paper focuses on the LoLa ontology, which formally describes DOL's vocabulary for logics, ontology languages (and their serialisations), as well as logic translations. Interestingly, to adequately formalise the logical relationships between these notions, LoLa itself needs to be axiomatised heterogeneously—a task for which we choose DOL. Namely, we use the logic RDF for ABox assertions, OWL for basic axiomatisations of various modules concerning logics, languages, and translations, FOL for capturing certain closure rules that are not expressible in OWL[4], and circumscription for minimising the extension of concepts describing default translations.

1 The Distributed Ontology Language (DOL) – Overview

An ontology in the Distributed Ontology Language (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.[5] Such links include (heterogeneous) *imports* and *alignments*, *conservative extensions* (important for the study of 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.

DOL is standardised within the OntoIOp (Ontology Integration and Interoperability) activity of ISO/TC 37/SC 3[6]. The international working group comprises around 50 experts (around 15 active contributors so far), representing

[4] For the sake of tool availability it is still helpful not to map everything to FOL.
[5] The languages that we call "basic" ontology languages here are usually limited to one logic and do not provide meta-theoretical constructs.
[6] TC = technical committee, SC = subcommittee

a large number of communities in ontological research and application, such as different (1) ontology languages and logics (e.g. Common Logic and OWL), (2) conceptual and theoretical foundations (e.g. model theory, proof theory), (3) technical foundations (e.g. ontology engineering methodologies and linked open data), and (4) application areas (e.g. manufacturing, bio-medicine, etc.). For details and earlier publications, see the project page at `http://ontoiop.org`.

The OntoIOp/DOL standard is currently in its final working draft stage and will be submitted as a committee draft (the first formal standardisation stage) in September 2012.[7] The final international standard ISO 17347 is scheduled for 2015. The standard specifies syntax, semantics, and conformance criteria:

Syntax: abstract syntax of distributed ontologies and their parts; three concrete syntaxes: a text-oriented one for humans, XML and RDF for exchange among tools and services, where RDF particularly addresses exchange on the Web. Here, we will use the DOL text syntax in listings but also explain the RDF vocabulary; for further details on the DOL syntaxes, see [6].

Semantics: (1) a *direct set-theoretical semantics* for the core of the language, extended by an *institutional and category-theoretic semantics* for advanced features such as ontology combinations (technically co-limits), where basic ontologies keep their original semantics; (2) a *translational semantics*, employing the semantics of the expressive Common Logic ontology language for all basic ontologies, taking advantage of the fact that for all basic ontology languages known so far translations to Common Logic have been specified or are known to exist[8]; (3) finally, there is the option of providing a *collapsed semantics*, where the semantics of the meta-theoretical language level provided by DOL (logically heterogeneous ontologies and links between them) is not just specified in semiformal mathematical textbook style, but once more formalised in Common Logic, thus in principle allowing for machine verification of meta properties. For details about the semantics, see [9].

Conformance criteria provide for DOL's extensibility to other basic ontology languages than those considered so far, including future ones. (1) A *basic ontology language* conforms with DOL if its underlying logic has a set-theoretic or, for the advanced features, an institutional semantics. Similar criteria apply to translations between languages. (2) A concrete syntax (*serialisation*) of a basic ontology language conforms if it supports IRIs (Unicode-aware Web-scalable identifiers) for symbols and satisfies further well-formedness criteria. (3) A *document* conforms if it is well-formed w.r.t. one of the DOL concrete syntaxes, which particularly requires explicitly mentioning all logics and translations employed. (4) An *application* essentially conforms if it is capable of processing conforming documents, and providing logical information that is implied by the formal semantics.

[7] The standard draft itself is not publicly available, but ISO/TC 37 has passed a resolution to make the final standard document open, as has been done with the related Common Logic standard [3].

[8] Even for higher-order logics this works, in principle, by using combinators.

2 A Graph of Logic Translations

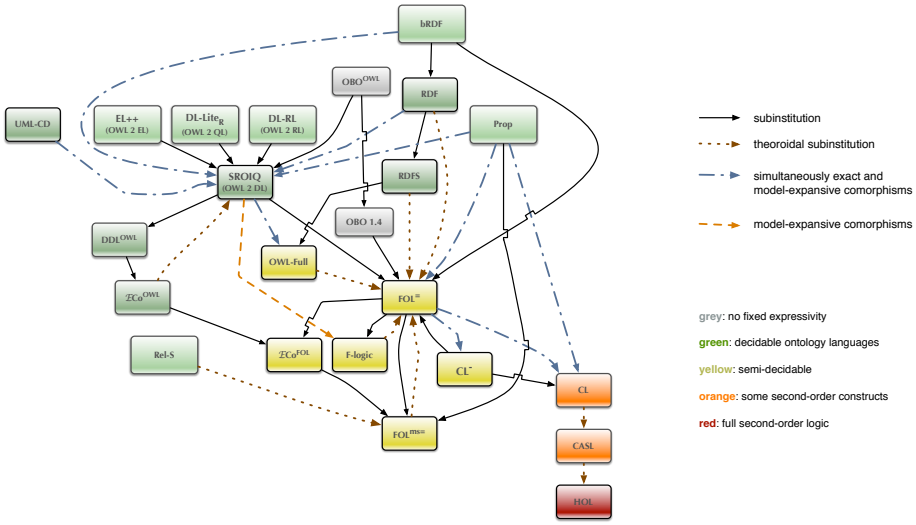

Fig. 1. The logic translation graph for the DOL-conforming languages

Fig. 1 is a revised and extended version of the graph of logics and translations introduced in [8]. 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 all of these logics in earlier publications [9, 8]. The definitions of the DOL-conformance of some central standard ontology languages and translations among them will be given as annexes to the standard, whereas the majority will be maintained in an open registry (cf. Sec. 3). Sec. 4 provides a more fine-grained view on translations (and projections).

3 A Registry for Ontology Languages and Mappings

The OntoIOp standard is not limited to a fixed set of ontology languages. It will be possible to use any (future) ontology language, logic, serialisation, or mapping (translation or projection) with DOL, once its conformance with the criteria specified in the standard has been established. This led to the idea of setting up a *registry* to which the community can contribute descriptions of any such languages, logics, serialisations, or mappings. In the current, early phase of the standardisation process, *we* are maintaining this registry manually. With the release of the final international standard, everyone will be able to make contributions, which an editorial board will review and approve or reject. Fig. 2

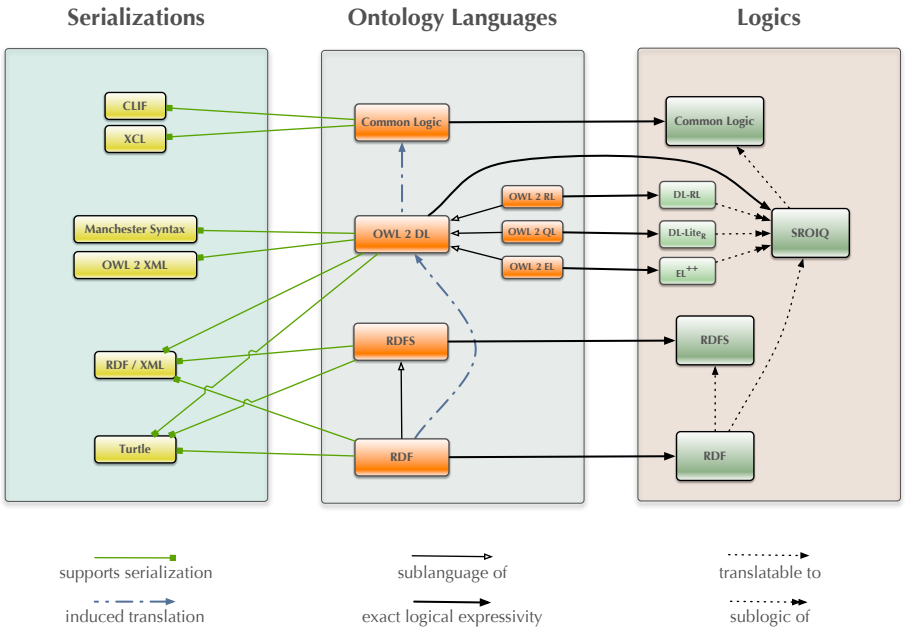

Fig. 2. Subset of the OntoIOp registry, shown as an RDF graph

shows a subset of the OntoIOp registry: a subgraph of Fig. 1 in the "logic" column, as well as related ontology languages and serialisations. Note that the relation between ontology languages and logics generally is *not* bijective: e.g. first-order logic is supported by various languages such as Common Logic, TPTP and CASL.

Any entry of the registry shall be identified by an IRI, so that DOL ontologies can refer to it. At these IRIs, when treated as URLs, there shall be a machine-readable description of the resource according to the linked data principles (cf. [5]), so that, e.g., any agent given a basic ontology can find out the languages this ontology can be translated into (cf. Sec. 6 for an example), or that for any language translation, its definition as well as implementations are available. The most widely supported format for linked data is RDF; we have realised the RDF vocabulary for the OntoIOp registry as a subset of the vocabulary used for serialising DOL ontologies as RDF.[9]

Starting with a plain RDFS vocabulary, we soon realised that we could deliver added value to tools supporting DOL by encoding additional information about the semantics of, e.g., translations into the vocabulary using some OWL constructs, and eventually arrived at a richer formalisation that goes beyond

[9] RDF only allows for *describing*, not for fully formally *defining* logics and translations. To that end, we are planning to alternatively offer full formalisations in the richer OMDoc language from the same IRIs.

OWL: the LoLa ontology. To realise the benefit of a machine-comprehensible representation of this semantics in a rich ontology language, consider DOL's understanding of an ontology language translation: Unless a direct translation on the language level has been specified (e.g. from Common Logic to CASL), one can translate an ontology from a language La to La' if the expressivity of these languages is exactly captured by two logics Lo and Lo', and Lo can (possibly transitively) be translated to Lo'. In a plain RDF graph, this would require multiple lookups.

4 Architecture of the LoLa Ontology

LoLa, the ontology of logics and languages, is implemented as a heterogeneous ontology in DOL, consisting of the following modules:

- An **OWL** core provides classes and properties for the basic concepts, including a basic axiomatisation.
- We use additional **FOL** axioms for closure rules not expressible in OWL, such as non-expressible role compositions.
- We use **circumscription** [7, 1] for minimising the extension of default translations.

The OntoIOp registry, is implemented as an RDF dataset, acting as the ABox of the LoLa ontology. The OntoIOp registry is available through a collection of linked data IRIs in the paths `http://purl.net/dol/{logics,languages,serializations,translations}`, e.g. `http://purl.net/dol/logics/SROIQ` for the logic $\mathcal{SROIQ}$. We made it originally available in RDF/XML, the most widely supported linked data format, but other syntaxes can be provided as well. It can be browsed with frontends like uriburner; try, e.g., `http://linkeddata.uriburner.com/about/html/http/purl.net/dol/logics/SROIQ`.

The OWL core of the LoLa ontology comprises classes for ontology languages, logics, mappings (translations or projections) between ontology languages and between logics, as well as serialisations. The LoLa properties relate all of the former classes to each other, as shown in Fig. 2, e.g. an ontology language to the serialisations that it supports, or to the logic that exactly formalises its expressivity, or an ontology language mapping to the logic mapping it has been derived from.

Fig. 3 shows the top-level classes of LoLa's OWL module, axiomatising logics, languages, and mappings to the extent possible in OWL. Concerning meta-level classes (that is, classes for describing the graph of languages and logics), Fig. 2 already has illustrated the interplay of ontology languages, logics and serialisations.

Object-level classes (that is, classes providing the vocabulary for expressing distributed ontologies) comprise ontologies, their constituents (namely entities, such as classes and object properties, and sentences, such as class subsumptions), as well as links between ontologies.

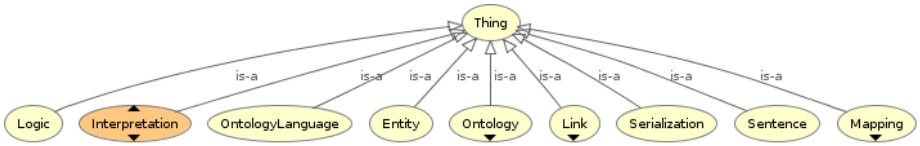

Fig. 3. Top-level classes in the OWL ontology

Mappings are modelled by a hierarchy of properties corresponding to the different types of edges in Fig. 1. For example, object properties such as `translatableTo` model the existence of a translation between two languages. `mappableToLanguage` models the fact that a language can be mapped to another one.

However, this only allows for covering the default translations between logics. E.g., we can express that the default translation from $\mathcal{SROIQ}$ to F-logic is a model-expansive comorphism. Besides further alternative translations that the community may contribute, there is, however, also another translation, which can be obtained from our graph, by composing the $\mathcal{SROIQ} \rightarrow$FOL$^=$ and FOL$^=\rightarrow$F-logic translations, resulting in a subinstitution. For expressing such alternatives, LoLa additionally reifies mappings into classes, whose hierarchy corresponds to that of the mapping properties.

Fig. 4 shows the inferred class hierarchy below the class `Mapping`, as computed within PROTÉGÉ. Notice that our ontology produces several cases of multiple inheritance. Mappings are split along the following dichotomies:

- *logic mapping* versus *ontology language mapping*, cf. Fig. 2.
- *translation* versus *projection*: a translation embeds or encodes an ontology 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 [4].
- *plain mapping* versus *simple theoroidal mapping* [4]: 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 roles etc.

Moreover, we have a class `DefaultMapping` for mappings that are assumed automatically as default when no mapping is given in a certain context.

Other classes concern the accuracy of the mapping, see [8] for details. These classes have mainly been introduced for the classification of logic mappings; however, via the correspondence between logics (mappings) and ontology languages (mappings), they apply to ontology languages as well. *Sublogics* are the most accurate mappings: they are just syntactic subsets. *Embeddings* come close to sublogics, like injective functions come close to subsets. If the model translation is surjective ("model expansion") or even bijective, the mapping is *faithful* in the sense that logical consequence is preserved and reflected, that is, inference

systems and 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 [2].

The full OWL ontology is available at `http://purl.net/dol/1.0/rdf#`; it serves, as said above, simultaneously as an RDF vocabulary for the linked dataset that constitutes the OntoIOp registry, and for serialising DOL ontologies in RDF—therefore the "rdf" name.

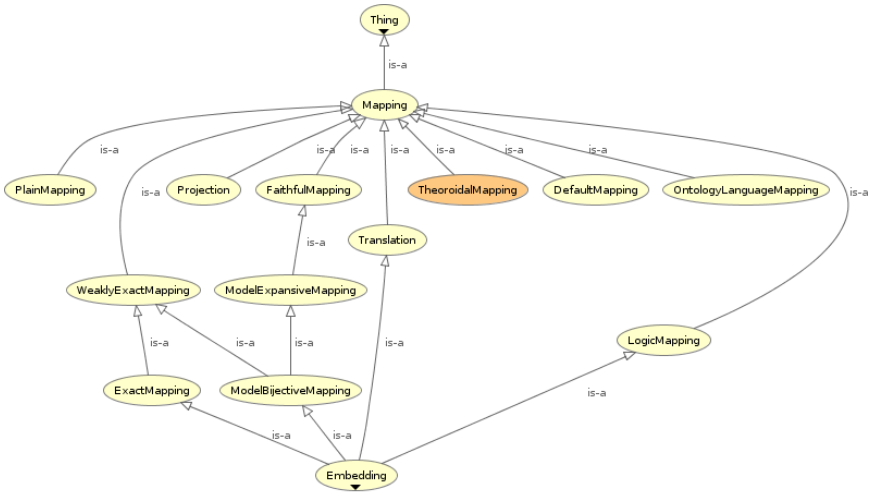

Fig. 4. The part of the OWL ontology concerning mappings

5 Putting It Together in DOL

DOL allows us to put together the pieces collected so far. First, we specify that the RDF registry conforms with the OWL ontology. This is achieved by projecting the registry from RDF to OWL[10], and stating an interpretation of theories of this into the OWL ontology.

We use circumscription [7, 1] for minimising the extent of the class `Default-Translation` and thus implementing a closed world assumption. This feature has been integrated into DOL in a logic independent way: in OWL, it has the effect that classes and object properties are minimised, while in first-order logic, extensions of predicates are.

[10] Basically, this projection turns the RDF graph into an OWL ABox. Impredicativity is escaped from by splitting names that are used in several roles into several distinct names.

Furthermore, we use first-order logic to formulate logical axioms that exceed the expressiveness of OWL. We here use the Common Logic Interchange Format (CLIF) [3]. One such axiom states that supported logics propagate along language translatability; see the ontology `LoLaRules` below.

```
%prefix( :      <http://purl.net/dol/>
         dol:   <http://purl.net/dol/1.0/rdf#>
         log:   <http://purl.net/dol/logics/>
         ser:   <http://purl.net/dol/serializations/>
         trans: <http://purl.net/dol/translations/> )%

distributed-ontology LoLa

%% projecting the RDF ABox to OWL
ontology ABox = registry hide along RDF2OWL end

%% TBox
ontology TBox = dol: end

%% the RDF registry conforms with the OWL ontology
interpretation conformant : ABox to TBox end

%% integrating RDF ABox with OWL TBox while minimising default mappings
logic log:OWL syntax ser:OWL/Manchester
ontology MinimizedABox =
  ABox and TBox
  minimize DefaultMapping %% circumscription-like minimisation
end

%% first-order rules for infering new facts in the registry
logic log:CommonLogic syntax ser:CommonLogic/CLIF
ontology LoLaRules =
  (forall (subLa superLa lo)
    (if (and (dol:translatableTo subLa superLa)
             (dol:mappableToLanguage subLa superLa)
             (dol:supportsLogic subLa lo))
        (dol:supportsLogic superLa lo)))
  ...
end

%% combining OWL ontology with first-order rules
logic log:CommonLogic syntax ser:CommonLogic/CLIF
ontology LoLa =
  dol: translate with OWL2CommonLogic
and
  LoLaRules
end
```

6 Using LoLa to Query the OntoIOp Registry

DOL-conforming applications can explore and query the OntoIOp registry to find out useful information about the logics and languages of concrete given ontologies, or about logics and languages in general.

The following query in the SPARQL RDF query language, e.g., returns all languages a given ontology is translatable to:

```
PREFIX dol: <http://purl.net/dol/1.0/rdf#>
SELECT DISTINCT ?target-language WHERE {
  # first determine, by querying the ontology itself, its language
  <http://ontohub.org/ontologies/my-ontology>
    dol:language ?theLanguageOfTheGivenOntology .
  # find out everything the language is translatable to
  ?theLanguageOfTheGivenOntology
    dol:translatableTo ?targetLanguage ;
    # just to be sure: We are only interested in mappings to languages.
    dol:mappableToLanguage ?targetLanguage .
  # (The use of two properties is owed to the orthogonal design of LoLa.)
}
```

This query assumes that both the information about the ontology and about the OntoIOp registry are available in RDF and ready to be queried as SPARQL. At the moment this cannot be taken for granted; however, we are working on Ontohub, an ontology repository engine, which we will, at the same time, also use to host the OntoIOp registry instead of the current static file deployment [6].

Aiming at wide tool support, the linked data graph that we deploy has all inferences of the LoLa ontology applied; this means in particular that, from a translation between two logics, it is inferred that the corresponding ontology languages are translatable into each other, and that the transitive closure of the translation graph has been computed. Therefore, the query shown above operates on a plain RDF graph, and the query engine does not have to have further inferencing support built in.

The following query focuses exclusively on the OntoIOp registry. It answers a frequent question in knowledge engineering: Which logic is the right one for formalising my conceptual model? For the sake of this example, we focus on knowledge *representability* and thus assume that a logic is suitable if it has translations from and to many other logics. This ignores questions of availability of reasoners for the respective logics, of tools performing the translations, and of their performance. Such information is not yet available in the OntoIOp registry itself, but could be compiled in a separate linked dataset that the registry would link to.

```
PREFIX dol: <http://purl.net/dol/1.0/rdf#>
SELECT ?logic, COUNT(?targetLogic) AS ?t, COUNT(?sourceLogic) AS ?s WHERE {
  ?logic a dol:Logic ;
         dol:translatableTo ?targetLogic ;
         dol:translatableFrom ?sourceLogic .
} ORDER BY ?t, ?s
```

7 Conclusion and Future Work

We have presented LoLa, an ontology of logics, languages, and mappings between them. This ontology formalises the semantics not only of these aspects of the Distributed Ontology Language DOL, but also of the vocabulary employed in the OntoIOp registry for extending the DOL framework with further logics, languages and mappings. LoLa is a heterogeneous ontology consisting of a core OWL module, which declares the vocabulary and provides a basic formalisation, a Common Logic module providing additional first-order rules; furthermore we employ DOL's logic-independent circumscription facility to minimise the extension of default translations. Along with our plans to publish not only machine-comprehensible descriptions of logics and mappings, but full formalisations, we will also expand LoLa to formalise further features of the DOL language, such as the vocabulary that describes the accuracy of a mapping (cf. Sec. 4).

Acknowledgements

The development of DOL is supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 "Spatial Cognition"; the first author is additionally supported by EPSRC grant EP/J007498/1. The authors would like to thank the OntoIOp working group within ISO/TC 37/SC 3 for their input, particularly Michael Grüninger for his advice regarding the hierarchy of types of ontology translations.

References

1. Bonatti, Piero A., Carsten Lutz, and Frank Wolter, 'The complexity of circumscription in dls', *J. Artif. Intell. Res. (JAIR)*, 35 (2009), 717–773.
2. Borzyszkowski, Tomasz, 'Logical systems for structured specifications', *Theoretical Computer Science*, 286 (2002), 197–245.
3. 'Common Logic (CL): a framework for a family of logic-based languages', Tech. Rep. 24707, ISO/IEC, 2007.
4. Goguen, Joseph, and Grigore Roşu, 'Institution morphisms', *Formal aspects of computing*, 13 (2002), 274–307.
5. Heath, Tom, and Christian Bizer, *Linked Data: Evolving the Web into a Global Data Space*, 1 edn., Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, San Rafael, CA, 2011.
6. Lange, Christoph, Till Mossakowski, Oliver Kutz, Christian Galinski, Michael Grüninger, and Daniel Couto Vale, 'The Distributed Ontology Language (DOL): Use cases, syntax, and extensibility', in *Terminology and Knowledge Engineering Conference (TKE)*, 2012.
7. McCarthy, John, 'Circumscription - a form of non-monotonic reasoning', *Artif. Intell.*, 13 (1980), 1–2, 27–39.
8. Mossakowski, Till, and Oliver Kutz, 'The Onto-Logical Translation Graph', in Oliver Kutz, and Thomas Schneider, (eds.), *Modular Ontologies*, IOS, 2011.
9. Mossakowski, Till, Oliver Kutz, and Christoph Lange, 'Three semantics for the core of the distributed ontology language', in *FOIS*, 2012.