# D3.2

**Extended Heterogeneous KR&R Framework for Concept Invention**

<table>
<thead>
<tr>
<th>Authors</th>
<th>Till Mossakowski, Mihai Codescu, Fabian Neuhaus, and Oliver Kutz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviewers</td>
<td>Marco Schorlemmer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grant agreement no.</th>
<th>611553</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>COINVENT - Concept Invention Theory</td>
</tr>
<tr>
<td>Date</td>
<td>May 11, 2015</td>
</tr>
<tr>
<td>Distribution</td>
<td>PU</td>
</tr>
</tbody>
</table>
Abstract

This deliverable specifies the extended KR&R infrastructure used for formalising conceptual blending within COINVENT. It includes an extensive description of the syntax and semantics of the Distributed Ontology Language DOL, with a special emphasis on heterogeneity and mappings. Moreover, we discuss current tool support.

Keyword list: Knowledge Representation, Reasoning, Distributed Ontology Language, DOL, Modularity, Interoperability, Institutions
Executive Summary

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. Also in model-driven engineering, there is a diversity of diagrams: UML consists of 15 different diagram types, and SysML provides further types. Finally, in software and hardware specification, a variety of formalisms are in use, like Z, VDM, first-order logic, temporal logic etc.

Another diversity appears at the level of ontology, model and specification modularity and relations among ontologies, specifications and models. There is ontology matching and alignment, module extraction, interpolation, ontologies linked by bridges, interpretation and refinement, and combination of ontologies, models and specifications.

The Distributed Ontology, Modeling and Specification Language (DOL) aims at providing a unified metalanguage for handling this diversity. In particular, DOL provides constructs for (1) "as-is" use of ontologies, models and specifications (OMS) formulated in a specific ontology, modelling or specification language, (2) OMS formalised in heterogeneous logics, (3) modular OMS, (4) mappings between OMS, and (5) networks of OMS. This paper sketches the design of the DOL language. DOL has been submitted as a proposal within the OntoIOp (Ontology, Model, Specification Integration and Interoperability) standardisation activity of the Object Management Group (OMG).

This deliverable plays an important role in the theoretical and practical development of the COINVENT project, as the DOL language is a central ingredient for the formalisation of blending diagrams and corresponding reasoning tasks. A recent example for this is published in [12]. This deliverable is a minor modification of a book chapter published in [35].
## Contents

1 Introduction .......................................................... 1

2 Foundations of the Distributed Ontology, Modeling and Specification Language (DOL) .................................................. 2
   2.1 Foundations ......................................................... 3
   2.2 Translations between Institutions .................................. 5
   2.3 A Graph of Logic Translations ...................................... 6

3 The Language DOL ........................................................ 7
   3.1 DOL Syntax and Semantics .......................................... 7
   3.2 Modular and Heterogeneous OMS .................................... 9
      3.2.1 Basic OMS ...................................................... 9
      3.2.2 IRI reference .................................................. 10
      3.2.3 Extension ....................................................... 10
      3.2.4 Union ........................................................... 11
      3.2.5 Translation ...................................................... 12
      3.2.6 Reduction ....................................................... 12
      3.2.7 Filtering ......................................................... 14
      3.2.8 Interpolation .................................................... 15
      3.2.9 Extraction ....................................................... 15
      3.2.10 Combination .................................................... 16
      3.2.11 Minimisation ................................................... 17
      3.2.12 Hide vs. Remove vs. Forget vs. Reject ...................... 18
   3.3 OMS Mappings ....................................................... 19
      3.3.1 Interpretation .................................................... 20
      3.3.2 Refinement ....................................................... 20
      3.3.3 Entailment ....................................................... 21
      3.3.4 OMS Equivalence ............................................... 22
      3.3.5 Conservative Extension ....................................... 22
      3.3.6 Alignment definition ......................................... 23
   3.4 Networks of OMS .................................................... 24
   3.5 Libraries of OMS .................................................... 25

4 Tool Support for DOL .................................................... 26

5 Conclusion and Future Work ........................................... 27
1 Introduction

Logical languages are used in several fields of computing for the development of formal, machine-processable texts that carry a formal semantics. Among those fields are 1) Ontologies formalising domain knowledge, 2) (formal) Models of systems, and 3) the formal Specification of systems. Ontologies, models and specifications will (for the purpose of this paper) henceforth be abbreviated as OMS.

An OMS provides formal descriptions which range in scope from domain knowledge and activities (ontologies, models) to properties and behaviours of hardware and software systems (models, specifications). While the use of OMS varies considerably, there are two recurring challenges: reusability and interoperability.

Reusability is an issue because the development of OMS is typically done manually by experts and, thus, an expensive process. Hence, it is desirable to be able to reuse existing OMS during the development of new OMS. This presupposes a framework that allows to build structured OMS by identifying modules and their relationships to each other. For example, it requires the ability to combine two existing OMS in a way that handles the namespaces of the OMS in an appropriate way. Further, the reuse of an existing OMS often requires that the OMS is adapted for its new purpose. For example, the adaption may require the extension of the OMS by new axioms, or the extraction of a subset of the OMS, or the change of its semantics from open world to closed world.

The interoperability challenge is closely related to the reusability challenge. Since the development of OMS is not an exact science and is usually driven by project specific requirements, two OMS that have been developed independently will represent the same domain in different and, often, conflicting ways. They may differ, for example, with respect to the terminology, or with respect to the definitions of the underlying concepts, or with respect to the perspective from which they represent their domain. Thus, in a situation where two independently developed OMS are supposed to be reused as modules of a larger OMS, the differences between these OMS will typically prevent them from working together properly. Overcoming this lack of interoperability may require an alignment or even an integration of these OMS. This typically involves the identification of synonyms, homonyms, and the development of bridge axioms, which connect the two OMS appropriately.

Both the reusability and the interoperability challenges are amplified by the diversity of OMS languages that are in use. For ontologies these include 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. Also in model-driven engineering, there is a diversity of diagrams: UML consists of 15 different diagram types, and SysML provides further types. Finally, in software and hardware specification, a variety of formalisms are in use, like Z, VDM, first-order logic, temporal logic etc. These languages do not just differ with respect to their syntax, but with respect to their semantics and to their levels of expressiveness.

To address both challenges we propose the Distributed Ontology, Modeling and Specification Language (DOL). DOL is a metalanguage that enables the reuse, integration, and alignment of existing OMS – even if they are written in different formalisms. The underlying methodological stance is that it would be futile to attempt to develop yet another OMS language that would subsume all the others; instead we have to accept the diversity of OMS languages and the diversity of
perspectives that are represented by different OMS. DOL provides a sound and formal semantic basis for specifying structured OMS, which may reuse as modules several existing OMS (possibly written in different languages) without requiring any changes to these modules. Further, DOL allows to specify mappings between different OMS (e.g., alignments and logical entailments).

In particular, DOL enjoys the following distinctive features:

- modular OMS and OMS networks are specially supported,
- OMS can not only be aligned (as in BioPortal [42] and NeON [18]), but also combined along alignments,
- mappings between OMS (interpretation of theories, conservative extensions etc.) are supported,
- it supports a variety of OMS languages (OWL, RDF, Common Logic, first-order logic, CASL; planned: UML, relational database schema, F-logic, distributed description logics, and more),
- OMS can be translated to other OMS languages, and compared with OMS in other languages,
- heterogeneous OMS (i.e., structured OMS with modules written in different languages) can be built,
- OMS languages and OMS 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 OMS repository engine Ontohub, and Section 5 concludes.

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

The Distributed Ontology, Modeling and Specification Language (DOL) aims at providing a unified framework for (1) “as-is” use of OMS formulated in a specific OMS language, (2) modular OMS, (3) mappings between OMS, (4) OMS networks, and (5) OMS formalised in heterogeneous logics. Historically, the design of DOL has inherited many ideas and features (1) discussed in the Workshop on Modular Ontologies series [17, 16, 45, 25, 29, 47], (2) from the Alignment API [10], (3) from CLEAR, ASL and specifications in an arbitrary institution [5, 49, 43, 44], and (4) from the CASL (Common Algebraic Specification Language) and HetCASL (CASL’s heterogeneous extension) languages, standardised in IFIP WG 1.3 (Foundations of System Specification) [2, 31, 37, 26].

\[^1\]DOL has formerly been standardised within ISO/TC 37/SC 3. The OntoIOp (Ontology, Modelling and Specification Integration and Interoperability) activity is now being continued at OMG, see the project page at http://ontoiop.org.
\[^2\]See http://ifipwg13.informatik.uni-bremen.de
A library in DOL consists of modules formalised in basic OMS 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 OMS. DOL adds a meta-level on top, which allows for expressing heterogeneous OMS and mappings between OMS. Such mappings include (heterogeneous) imports and alignments, conservative extensions (important for studying OMS modules), and theory interpretations (important for reusing proofs). Thus, DOL gives OMS interoperability a formal grounding and makes heterogeneous OMS and services based on them amenable to automated verification. The basic syntax and semantics of DOL has been introduced in \cite{40,39}, and the general theory of heterogeneous specifications for OMS in \cite{28}. DOL uses internationalised resource identifiers (IRIs, the Unicode-aware superset of URIs) for all entities of OMS libraries 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 \cite{13}. 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 \cite{39} 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 main 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 the reduction of models against signature morphisms (generalising the notion of model reduct known from logic).

**Definition 1** An institution \cite{13} is a quadruple $I = (\text{Sign}, \text{Sen}, \text{Mod}, \models)$ consisting of the following:

- a category $\text{Sign}$ of signatures and signature morphisms,
- a functor $\text{Sen} : \text{Sign} \to \text{Set}$ giving, for each signature $\Sigma$, the set of sentences $\text{Sen}(\Sigma)$, and for each signature morphism $\sigma : \Sigma \to \Sigma'$, the sentence translation map $\text{Sen}(\sigma) : \text{Sen}(\Sigma) \to \text{Sen}(\Sigma')$, where often $\text{Sen}(\sigma)(\phi)$ is written as $\sigma(\phi)$.

\footnote{3The languages that we call “basic” OMS languages here are usually limited to one logic and do not provide meta-theoretical constructs.}

\footnote{4$\text{Set}$ is the category having all sets as objects and functions as arrows.}
• a functor \( \text{Mod} : \text{Sign}^{op} \to \text{Cat} \) giving, for each signature \( \Sigma \), the category of models \( \text{Mod}(\Sigma) \), and for each signature morphism \( \sigma : \Sigma \to \Sigma' \), the reduct functor \( \text{Mod}(\sigma) : \text{Mod}(\Sigma') \to \text{Mod}(\Sigma) \), where often \( \text{Mod}(\sigma)(M') \) is written as \( M'|\sigma \), and \( M'|\sigma \) is called the \( \sigma \)-reduct of \( M' \), while \( M' \) is called a \( \sigma \)-expansion of \( M'|\sigma \),

• a satisfaction relation \( |_\Sigma \subseteq |\text{Mod}(\Sigma)| \times \text{Sen}(\Sigma) \) for each \( \Sigma \in |\text{Sign}| \), such that for each \( \sigma : \Sigma \to \Sigma' \) in \( \text{Sign} \) the following satisfaction condition holds:

\[
(*) \quad M' \models_\Sigma \sigma(\varphi) \iff M'|\sigma \models \varphi
\]

for each \( M' \in |\text{Mod}(\Sigma')| \) and \( \varphi \in \text{Sen}(\Sigma) \).

It is also possible to complement an institution with a proof theory, introducing a derivability relation between sentences, formalised as an entailment system \[34\]. 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, object and data properties. OWL signature morphisms map classes to classes, individuals to individuals, object properties to object properties and data properties to data 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.

Strictly speaking, this institution captures OWL 2 DL without restrictions in the sense of \[46\]. The reason is that in an institution, the sentences can be used for arbitrary formation of theories. This is related to the presence of DOL’s union operator on OMS. OWL 2 DL’s specific restrictions on theory formation can be modelled inside this institution, as a constraint on OMS. This constraint is generally not preserved under unions or extensions. DOL’s multi-logic capability allows the clean distinction between ordinary OWL 2 DL and OWL 2 DL without restrictions.

In this framework, a basic OMS \( O \) 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 OMS \( O \), we denote by \( \text{Sig}(O) \) the signature of the OMS. An OMS 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 \( E_2 \). Here, in an arbitrary institution, notions like logical consequence, satisfiability etc. can be defined in the standard way.

In the following we will need to assume existence of inclusions between signatures and of their unions. These concepts can be captured in a categorical setting using inclusion systems \[11\]. However, inclusion systems are too strong for our purposes and therefore we will work under weaker assumptions.

\(^5\text{Cat} \) is the category of categories and functors. Strictly speaking, \( \text{Cat} \) is not a category but only a so-called quasi-category, which is a category that lives in a higher set-theoretic universe.
Definition 2 An inclusive category is a category having a broad subcategory which is a partially ordered class with finite products and coproducts, called intersection (denoted \( \cap \)) and union (denoted \( \cup \)) such that for each pair of objects \( A, B \), \( A \cup B \) is a pushout of \( A \cap B \) in the category.

A category has pushouts which preserve inclusions iff there exists a pushout

\[
\begin{array}{ccc}
A & \longrightarrow & A' \\
\downarrow & & \downarrow \\
B & \longrightarrow & B'
\end{array}
\]

for each span where one arrow is an inclusion.

A functor between two inclusive categories is inclusive if it takes inclusions in the source category to inclusions in the target category.

Definition 3 An institution is weakly inclusive if

- \( \text{Sign} \) is inclusive and has pushouts which preserve inclusions,
- \( \text{Sen} \) is inclusive, and
- each model category have a broad subcategory of inclusions.

Let \( I \) be a weakly inclusive institution. We say that \( I \) has differences, if there is a binary operation \( \setminus \) on signatures, such that for each pair of signatures \( \Sigma_1, \Sigma_2 \), we have:

1. \( \Sigma_1 \setminus \Sigma_2 \subseteq \Sigma_1 \)
2. \( (\Sigma_1 \setminus \Sigma_2) \cap \Sigma_2 = \emptyset \)
3. for any \( \Sigma \) with the properties 1. and 2. above, \( \Sigma \subseteq \Sigma_1 \setminus \Sigma_2 \).

2.2 Translations between Institutions

Several notions of translations between institutions can be introduced. The most frequently used variant are institution comorphisms. 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.

Definition 4 An institution comorphism from an institution \( I = (\text{Sign}^I, \text{Mod}^I, \text{Sen}^I, |^I) \) to an institution \( J = (\text{Sign}^J, \text{Mod}^J, \text{Sen}^J, |^J) \) consists of a functor \( \Phi : \text{Sign}^I \longrightarrow \text{Sign}^J \), and two natural transformations \( \beta : \text{Mod}^J \circ \Phi \longrightarrow \text{Mod}^I \) and \( \alpha : \text{Sen}^J \longrightarrow \text{Sen}^I \circ \Phi \), such that

\[
M' |^J_{\Phi(\Sigma)} \iff \beta\Sigma(\varphi) \iff \alpha\Sigma(M') |^I \varphi.
\]

holds, called the satisfaction condition.

\[^6\]That is, with the same objects as the original category.
Here, $\Phi(\Sigma)$ is the translation of the signature $\Sigma$ from institution $I$ to institution $J$, $\alpha_\Sigma(\varphi)$ is the translation of the $\Sigma$-sentence $\varphi$ to a $\Phi(\Sigma)$-sentence, and $\beta_\Sigma(M')$ is the translation (or perhaps better: reduction) of the $\Phi(\Sigma)$-model $M'$ to a $\Sigma$-model. The naturality of $\alpha$ and $\beta$ mean that for each signature morphism $\sigma: \Sigma \to \Sigma'$ in $I$ the following squares commute:

\[
\begin{array}{ccc}
\text{Sen}^I(\Sigma) & \xrightarrow{\alpha_\Sigma} & \text{Sen}^J(\Phi(\Sigma)) \\
\downarrow & & \downarrow \\
\text{Mod}^I(\Sigma) & \xrightarrow{\beta_\Sigma} & \text{Mod}^J(\Phi(\Sigma))
\end{array}
\]

\[
\begin{array}{ccc}
\text{Sen}^I(\Sigma') & \xrightarrow{\alpha_{\Sigma'}} & \text{Sen}^J(\Phi(\Sigma')) \\
\downarrow & & \downarrow \\
\text{Mod}^I(\Sigma) & \xrightarrow{\beta_\Sigma} & \text{Mod}^J(\Phi(\Sigma))
\end{array}
\]

**Definition 5** An institution morphism from an institution $I = (\text{Sign}^I, \text{Mod}^I, \text{Sen}^I, \models^I)$ to an institution $J = (\text{Sign}^J, \text{Mod}^J, \text{Sen}^J, \models^J)$ consists of a functor $\Phi: \text{Sign}^I \to \text{Sign}^J$, and two natural transformations $\beta: \text{Mod}^I \Rightarrow \text{Mod}^J \circ \Phi$ and $\alpha: \text{Sen}^J \circ \Phi \Rightarrow \text{Sen}^I$, such that

$$M \models^I_\Sigma \alpha_\Sigma(\varphi) \iff \beta_{\Phi(\Sigma)}(M) \models^J_{\Phi(\Sigma)} \varphi.$$ holds, called the satisfaction condition.

Mappings of institutions 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). It is an interesting informal observation that translations can be formalised as institution comorphisms, and projections as institution morphisms.

- **plain mapping versus simple theoroidal mapping** [14]: 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.

Mappings can also be classified according to their accuracy; see [38] 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 OMS structuring operations [4].

### 2.3 A Graph of Logic Translations

Figure 1 is a revised and extended version of the graph of logics and translations introduced in [38]. 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 [40, 38]. The
definitions of the DOL conformance of some central standard OMS 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.

3 The Language DOL

3.1 DOL Syntax and Semantics

The DOL language is not “yet another OMS language”, but a metalanguage for expressing rel-
lations between OMS. Therefore, any OMS written in any conforming OMS language also is a
DOL OMS. Therefore, when working with DOL users can reuse OMS as they are, no changes are
required.

DOL provides abstract syntax categories for:

1. OMS (ontologies, models and specifications). Basic OMS are OMS that are written in some
OMS language (e.g., OWL or CASL). A modular or structured OMS is written in a modular
way, with the help of DOL structuring operations. A heterogeneous OMS is a modular
OMS that involves modules, which are written in different OMS languages. The semantics
of OMS is given by a signature and a class of models. In some cases, we can additionally
provide a theory-level semantics of OMS, 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 OMS flattenable if it has a theory-level semantics
and elusive if it only admits a model-level semantics. Whether an OMS is flattenable can be decided based on the structuring operations on OMS, as follows:

**Flattenable OMS**: basic OMS are flattenable; if all their components OMS are flattenable, then the following operations on OMS yield flattenable OMS: extension, union, translation, interpolate/forget, extract, reference, qualification, combination.

**Elusive OMS**: the reduction, minimisation, or maximisation of an OMS is elusive; further, any OMS containing an elusive OMS is elusive.\(^7\)

For detailed definitions of these types of OMS, see Section 3.2.

2. **OMS mappings**. They denote relations between two OMS or OMS networks, typically along a signature morphism. Some mappings may also involve other OMS or other signatures. Examples of OMS mappings are interpretations (specifying a logical consequence relationship between OMS), equivalences of OMS (specifying that their model classes are in bijective correspondence), conservative extensions (between OMS and their modules), OMS alignment. They are presented in Section 3.3.

3. **Networks** of OMS. Networks are graphs with nodes labelled with OMS and edges labelled with OMS mappings. The edges show how two OMS are interlinked. The rationale behind networks is that they provide a way to specify or model complex distributed systems (or domains), where a single OMS would become too complex (this especially can be the case if the OMS are formulated in different OMS languages). Instead, the different OMS of the network provide different viewpoints on the system, while their compatibility is ensured via mappings. Networks are discussed in Section 3.4.

4. **Libraries** of OMS. OMS, mappings and networks are organised in libraries. A library consists of a list of declarations involving (possibly modular and/or heterogeneous) OMS. These declarations can be definitions (assigning a name to an OMS, OMS mapping or network of OMS) and qualifications of the current language, logic and/or serialisation. This is detailed in Section 3.5.

The semantics of DOL is based on a fixed (but in principle arbitrary) logic graph. A logic graph is given by a collection of institutions, institution morphisms and institution comorphisms (serving as logics, logic reductions and logic translations). Moreover, some of the institution comorphisms are marked as default translations and some of the institution morphisms are marked as default projection (but only at most one between a given source and target institution).

We assume that for each institution in the logic graph there is a trivial signature \(\emptyset\) with model class \(\emptyset\) and such that there exists a unique signature morphism from \(\emptyset\) to any signature of the institution. Moreover we assume the existence of a designated error logic in the graph, and a partial union operation on logics, denoted \(\bigcup\): \(L_1 \bigcup L_2 = (L, \rho_1 : L_1 \rightarrow L, \rho_2 : L_2 \rightarrow L)\), when defined.

\(^7\)Note that extension, union, translation, reference, qualification and combination are defined for flattenable and elusive OMS, while interpolate/forget and extract are only defined for flattenable OMS.
3.2 Modular and Heterogeneous OMS

Modular and heterogeneous OMS are generated by the following grammar, where $\Sigma$ is a signature, $\Delta$ is a set of sentences over $\Sigma$, $\sigma$ a signature morphism, $I$ an institution, $\rho$ an institution comorphism and $\mu$ an institution morphism:

\[
\text{OMS} := \langle I, \Sigma, \Delta \rangle | \text{IRI} | \text{OMS and OMS} | \text{OMS then OMS} | \text{OMS with } \sigma | \text{OMS with translation } \rho | \text{OMS reveal } \Sigma | \text{OMS hide } \Sigma | \text{OMS hide along } \mu | \text{OMS keep } \Sigma | \text{OMS keep } I | \text{OMS forget } \Sigma | \text{OMS extract } \Sigma | \text{OMS remove } \Sigma | \text{OMS select } \langle \Sigma, \Delta \rangle | \text{OMS reject } \langle \Sigma, \Delta \rangle | \text{minimize OMS} | \text{maximize OMS} | \text{combine Network}
\]

The semantics of an OMS $O$ has four components:

- the institution of $O$, denoted $\text{Inst}(O)$,
- the signature of $O$, denoted $\text{Sign}(O)$ (which is a signature in $\text{Inst}(O)$),
- the models of $O$, denoted $\text{Mod}(O)$ (which is a class of models over $\text{Sign}(O)$),
- the axioms of $O$, denoted $\text{Ax}(O)$ (which is a set of sentences over $\text{Sign}(O)$).

For elusive OMS, $\text{Ax}(O)$ is undefined. For flattenable OMS, $\text{Mod}(O)$ can be obtained as $\text{Mod}(O) = \{ M \in \text{Mod}(\text{Sign}(O)) | M \models \text{Ax}(O) \}$.

In the following we discuss the different kinds of (possibly modular and/or heterogeneous) OMS.

3.2.1 Basic OMS

A basic OMS $O$ written inline, in a conforming OMS language and serialisation. The semantics is inherited from the OMS language $I$ and results in a theory $\langle \Sigma, \Delta \rangle$ (therefore, for simplicity, in the syntax above, we have identified the basic OMS with $\langle I, \Sigma, \Delta \rangle$). $O$ can also be an OMS fragment, which means that some of the symbols or axioms may refer to symbols declared outside $O$ (i.e. in an imported OMS). This is mainly used for extensions and equivalences. Here are two sample ontologies in OWL (using Manchester syntax) and Common Logic (using CLIF):

\begin{verbatim}
Class: Woman EquivalentTo: Person and Female
ObjectProperty: hasParent
\end{verbatim}

---

8This is a mathematically abstracted version of DOL. In reality, signatures are represented by symbol sets, and signature morphisms by symbol maps. The details of passing from symbol sets (resp. maps) to signatures (resp. signature morphisms) are left out here. Also, we have left out OMS bridges, since their design is still being discussed.

9The theory of $O$, written $\text{Th}(O)$, is the closure of $\text{Ax}(O)$ under logical entailment. Note, however, that throughout the text we use ‘theory’ also more informally as denoting some set of axioms in a particular signature and logic.

10$I$ is normally determined by the context of the enclosing library and passed around as an additional parameter of the semantics. For simplicity, here we let $I$ become part of the basic OMS.
(cl-module PreOrder
  (forall (x) (le x x))
  (forall (x y z) (if (and (le x y) (le y z)) (le x z))))

Formally,

- $\text{Inst}(I, \Sigma, \Delta) = I$
- $\text{Sign}(I, \Sigma, \Delta) = \Sigma$
- $\text{Mod}(I, \Sigma, \Delta) = \{ M \in \text{Mod}(\Sigma) \mid M \models \Delta \}$
- $\text{Ax}(I, \Sigma, \Delta) = \Delta$.

3.2.2 IRI reference

An IRI reference to an OMS existing on the Web\footnote{Note that not all OMS can be downloaded by dereferencing their IRIs. Implementing a catalogue mechanism in DOL-aware applications might remedy this problem.} possibly abbreviated using prefixes\footnote{Some of the following listings abbreviate IRIs using prefixes but omit the prefix bindings for readability.} For example:

<http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/pizza.owl>

or alternatively

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

The semantics of such an IRI reference would require a global environment mapping IRIs to (semantics of) OMS. However, for simplicity, we omit the global environment (and therefore also the semantics of IRI references) here.

3.2.3 Extension

An extension of an OMS by new symbols and axioms, written $O_1$ then $O_2$, where $O_2$ is an OMS (fragment) in a conforming OMS language. The resulting signature is that of $O_1$, augmented with the symbols in $O_2$. A model of an extension OMS 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 $O_1$ (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

The semantics of $O = O_1$ then $O_2$ is

- $\text{Inst}(O) = \text{Inst}(O_1) \cup \text{Inst}(O_2)$
- $\text{Sign}(O) = \text{Sign}(O_1) \cup \text{Sign}(O_2)$
- $\text{Mod}(O) = \{ M \in \text{Mod}(\text{Sign}(O)) \mid M|_{\text{Sign}(O_i)} \in \text{Mod}(O_i), \text{ for } i = 1, 2 \}$
- $\text{Ax}(O) = \text{Ax}(O_1) \cup \text{Ax}(O_2)$

where $O_2$ is analysed in the context of previous declarations in $O_1$, as indicated by adding an index in its semantics.

3.2.4 Union

A union of two self-contained OMS (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 OMS is a particularly common operation in the RDF logic, where it is known as merging graphs [19 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:

{ :OVGU rdfs:label "Otto-von-Guericke-Universität Magdeburg" @de . } and
{ :OVGU geo:lat "52.1403" ^xsd:float . } and
{ :OVGU owl:sameAs <http://de.dbpedia.org/page/OvGU> . }

The semantics of $O = O_1$ and $O_2$ is

- $\text{Inst}(O) = I$ where $\text{Inst}(O_1) \cup \text{Inst}(O_2) = (I, (\Phi_1, \alpha_1, \beta_1) : \text{Inst}(O_1) \rightarrow I, (\Phi_2, \alpha_2, \beta_2) : \text{Inst}(O_2) \rightarrow I)$
- $\text{Sign}(O) = \Phi_1(\text{Sign}(O_1)) \cup \Phi_2(\text{Sign}(O_2))$
- $\text{Mod}(O) = \{ M \in \text{Mod}(\text{Sign}(O)) \mid \beta_2(M|_{\text{Sign}(O_i)}) \in \text{Mod}(O_i), \text{ for } i = 1, 2 \}$
- $\text{Ax}(O) = \alpha_1(\text{Ax}(O_1)) \cup \alpha_2(\text{Ax}(O_2))$.

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.
3.2.5 Translation

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

```plaintext
logic OWL : {
  ObjectProperty: isProperPartOf
  Characteristics: Asymmetric
  SubPropertyOf: isPartOf }
with translation OWL22CommonLogic
then
  (if (and (isProperPartOf x y) (isProperPartOf y z)) (isProperPartOf x z))
```

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

The semantics of $O = O'$ with $\sigma$ is

- $\text{Inst}(O) = \text{Inst}(O')$
- $\text{Sign}(O) = \Sigma'$ where $\sigma : \text{Sign}(O') \rightarrow \Sigma'$
- $\text{Mod}(O) = \{ M \in \text{Mod}(\Sigma') | M|_{\sigma} \in \text{Mod}(O') \}$
- $\text{Ax}(O) = \sigma(\text{Ax}(O'))$.

The semantics of $O = O'$ with translation $\rho$ is

- $\text{Inst}(O) = I$, where $\rho = (\Phi, \alpha, \beta) : \text{Inst}(O') \rightarrow I$
- $\text{Sign}(O) = \Phi(\text{Sign}(O))$
- $\text{Mod}(O) = \{ M \in \text{Mod}(\text{Sign}(O)) | \beta_{\text{Sign}(O)}(M) \in \text{Mod}(O') \}$
- $\text{Ax}(O) = \alpha_{\text{Sign}(O)}(\text{Ax}(O'))$.

3.2.6 Reduction

A *reduction* of an OMS to a smaller signature $\Sigma$ is written $O \text{ reveal } \Sigma$. Alternatively, it can be written $O \text{ hide } \Sigma$, where $\Sigma$ is the set of symbols to be hidden (i.e. this is equivalent to $O \text{ reveal } \text{Sig}(O) \setminus \Sigma$). 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,

```plaintext
sort Elem
ops 0: Elem; _+_: Elem * Elem -> Elem; inv: Elem -> Elem
forall x, y, z : Elem
  . 0 + x = x
  . x + (y + z) = (x + y) + z
  . x + inv(x) = 0
reveal Elem, 0, _+_
```
revealing everything except the inverse operation \( \text{inv} \) 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:

```plaintext
ontology Pizza = \% a simplified remake of the Pizza ontology
    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 \( O \ \text{hide along} \ \mu \), 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 \( \mathcal{SROIQ} \)) vs. RDF is a typical language combination:

```plaintext
ontology TBoxABox =
    Pizza hide along OWL22RDF
then logic RDF : {
    :myPizza :hasTopping
    [ a :TomatoTopping ], [ a :MozzarellaTopping ] .
}
```

The semantics of \( O = O' \ \text{reveal} \ \Sigma' \)

\begin{itemize}
  \item \( \text{Inst}(O) = \text{Inst}(O') \)
  \item \( \text{Sign}(O) = \Sigma' \)
  \item \( \text{Mod}(O) = \{ M | M \in \text{Mod}(O') \} \) where \( t: \Sigma' \to \text{Sign}(O') \) is the inclusion
  \item \( \text{Ax}(O) \) is undefined.
\end{itemize}

The semantics of \( O = O' \ \text{hide} \ \Sigma' \)

\begin{itemize}
  \item \( \text{Inst}(O) = \text{Inst}(O') \)
  \item \( \text{Sign}(O) = \text{Sign}(O') \setminus \Sigma' \)
  \item \( \text{Mod}(O) = \{ M | M \in \text{Mod}(O') \} \) where \( t: \text{Sign}(O') \setminus \Sigma' \to \text{Sign}(O') \) is the inclusion
  \item \( \text{Ax}(O) \) is undefined.
\end{itemize}

The semantics of \( O = O' \ \text{hide along} \ \mu \)

\begin{itemize}
  \item \( \text{Inst}(O) = \text{Inst}(O') \)
  \item \( \text{Sign}(O) = \text{Sign}(O') \setminus \Sigma' \)
  \item \( \text{Mod}(O) = \{ M | M \in \text{Mod}(O') \} \) where \( t: \text{Sign}(O') \setminus \Sigma' \to \text{Sign}(O') \) is the inclusion
  \item \( \text{Ax}(O) \) is undefined.
\end{itemize}
• \( \text{Inst}(O) = I \) where \( \mu = (\Phi, \alpha, \beta) : \text{Inst}(O) \rightarrow I \)

• \( \text{Sign}(O) = \Phi(\text{Sign}(O')) \)

• \( \text{Mod}(O) = \{ \beta_{\text{Sign}(O')}(M) \mid M \in \text{Mod}(O') \} \)

• \( \text{Ax}(O) \) is undefined.

### 3.2.7 Filtering

A filtering \( O \) select \((\Sigma, \Delta)\), which selects those sentences from \( O \) that have signature \( \Sigma \), plus those in \( \Delta \) (where \( \Delta \) is a subset \( \text{Ax}(O) \)). It can also be written \( O \) reject \((\Sigma, \Delta)\), where \( \Sigma \) is the set of symbols and \( \Delta \) the set of axioms to be hidden. For example, we can select all axioms of Galen involving Drugs, Joints, or Bodyparts by:

```log
logic OWL
ontology myGalenn =
  <http://example.org/GALEN/galen.owl>
  select Drugs, Joints, Bodyparts
end
```

The semantics of \( O = O' \) select \((\Sigma, \Delta)\) is defined only if \( \Sigma \subseteq \text{Sign}(O) \) and \( \Delta \subseteq \text{Ax}(O) \), and in that case, it is given by

• \( \text{Inst}(O) = \text{Inst}(O') \)

• \( \text{Sign}(O) = \Sigma' \) where \( \Sigma' \) is the smallest signature with \( \Sigma \subseteq \Sigma' \) and \( \Delta \subseteq \text{Sen}(\Sigma) \)

• \( \text{Ax}(O) = (\text{Ax}(O') \cap \text{Sen}(\text{Sign}(O))) \cup \Delta \)

• \( \text{Mod}(O) \) is the class of all \( \text{Ax}(O) \)-models.

The semantics of \( O = O' \) reject \((\Sigma, \Delta)\) is

• \( \text{Inst}(O) = \text{Inst}(O') \)

• \( \text{Sign}(O) = \text{Sign}(O') \setminus \Sigma \)

• \( \text{Ax}(O) = \text{Ax}(O') \cap \text{Sen}(\text{Sign}(O)) \setminus \Delta \)

• \( \text{Mod}(O) \) is the class of all \( \text{Ax}(O) \)-models.

---

\(^{14}\)We assume that GALEN is available as an OWL ontology.

\(^{15}\)If this smallest signature does not exist, the semantics is undefined.
3.2.8 Interpolation

An approximation (or technically, uniform interpolation) of an OMS, either in a subsignature or a sublogic (written $O \text{ keep } \Sigma$, $O \text{ keep } \Sigma \text{ keep } L$ or $O \text{ keep } L$, where $\Sigma$ is a signature and $L$ is a logic). The effect is that sentences not expressible in $\Sigma$ (resp. $L$) are weakened or removed, but the resulting theory still has the same consequences, as far as these are expressible in $\Sigma$ (and/or $L$). Technically, this is a uniform interpolant \[48, 33\]. For example, we can interpolate the first-order DOLCE mereology in OWL\[16\]

$$\text{DOLCE}_\text{Mereology} \text{ keep } \text{OWL}$$

Dually, $O \text{ forget } \Sigma$ or $O \text{ forget } \Sigma \text{ keep } L$ interpolates $O$ with the signature $\text{Sig}(O) \setminus \Sigma$, i.e. $\Sigma$ specifies the symbols that need to be left out (and optionally, $L$ specifies a sublogic that needs to be targeted). Cf. the notion of forgetting in \[48, 33\]. For example,

$$\text{Pizza forget VegetarianTopping}$$

This has both a model-theoretic and a theory-level semantics, i.e., it yields a theory in the reduced signature (without VegetarianTopping). In contrast, Pizza hide VegetarianTopping has only a model-level semantics (see also the comparison in section 3.2.12).

The semantics of $O = O' \text{ keep } \Sigma \text{ keep } I$ is

- **Inst**($O$) = $I$ and $(\Phi, \alpha, \beta) : \text{Inst}(O') \rightarrow I$ is the default projection (in case $I$ is missing, it is the identity on Inst($O'$))
- **Sign**($O$) = $\Phi(\Sigma)$
- **Ax**($O$) = $\alpha^{-1}_{\text{Sign}(O')} (\text{Ax}(O')^*) \cap \text{Sen}^I(\text{Sign}(O))$\[17\] i.e. that part of Ax($O'$) that can be expressed in the smaller signature and logic
- **Mod**($O$) is the class of Ax($O$)-models

The semantics of $O \text{ forget } \Sigma' \text{ keep } I$ is the same as the semantics of

$O \text{ keep } (\text{Sign}(O) \setminus \Sigma') \text{ keep } I$.

3.2.9 Extraction

A module extracted from an OMS, written $O \text{ extract } \Sigma$, where $\Sigma$ is a sub-signature of Sig($O$). The extracted module is a subOMS of $O$ with signature larger than (or equal to) $\Sigma$, such that $O$ is a conservative extension of the extracted module. Intuitively, a module (in the sense of module extraction) is a small sub-OMS that says the same about $\Sigma$ as the OMS $O$ itself. For example, we can extract from GALEN a module referring to drugs, joints and body parts:

```
logic OWL
ontology myGalene =
```

\[16\] Interpolants need not always exist, and even if they do, tools might only be able to approximate them.

\[17\] In practice, one looks for a finite subset that still is logically equivalent to this set. Note that $\Delta^*$ is the set of logical consequences of $\Delta$, i.e. $\Delta^* = \text{Th}(\Delta)$. 

15
The semantics of $O = O'$ extract $\Sigma$ is

- $\text{Inst}(O) = \text{Inst}(O')$
- $\text{Sign}(O) = \Sigma \cup \Sigma'$
- $\text{Ax}(O) = \Delta'$
- $\text{Mod}(O)$ is the class of $\text{Ax}(O)$-models

where $\langle \Sigma', \Delta' \rangle$ is the smallest depleting $\Sigma$-module \[^{18}\]\[^{19}\], i.e. the smallest sub-theory $\langle \Sigma', \Delta' \rangle$ of $\langle \text{Sign}(O'), \text{Ax}(O') \rangle$ such that the following model-theoretic inseparability holds

$$\text{Ax}(O') \setminus \Delta' \equiv_{\Sigma \cup \Sigma} \emptyset.$$ 

This means intuitively that $\text{Ax}(O') \setminus \Delta'$ cannot be distinguished from $\emptyset$ (what $\Sigma' \cup \Sigma$ concerns) and formally that

$$\{M|_{\Sigma \cup \Sigma} | M \in \text{Mod}(\text{Sign}(O')) \cdot M \models \text{Ax}(O') \setminus \Delta'\} = \{M|_{\Sigma \cup \Sigma} | M \in \text{Mod}(\text{Sign}(O'))\}.$$ 

Dually, $O$ remove $\Sigma$ extracts w.r.t. the signature $\text{Sign}(O') \setminus \Sigma$, i.e. the semantics is given by that of $O$ extract $\text{Sig}(O) \setminus \Sigma$.

### 3.2.10 COMBINATION

A combination of OMS, written combine $N$, where $N$ is a network. The simplest example of a combination is a disjoint union (we here translate OWL OMS into many-sorted OWL in order to be able to distinguish between different universes of individuals):

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

ontology Publications2 =
    Class: Thing
    Class: Article SubClassOf: Thing
```

\[^{18}\]If the smallest such sub-theory does not exist, the semantics is undefined. In \[^{23}\], it is shown that it does exist in usual institutions.

\[^{19}\]Note that the resulting module can still contain symbols from $\Sigma$, because the resulting signature may be enlarged.
logic MS-OWL

network Publications_Network =
1 : Publications1 with translation OWL2MS-OWL,
2 : Publications2 with translation OWL2MS-OWL
end

ontology Publications_Combined =
combine
Publications_Network
%% implicitly: Article \mapsto 1:Article ...
%% Article \mapsto 2:Article ...
end

If mappings or alignments are present, the semantics of a combination is a quotient of a disjoint union (symbols related along the edges are identified). Technically, this is a colimit, see \cite{50,7}.

An example for this is given along with the examples for alignments below.

The semantics of $O = \text{combine } N$ is

1. $\text{Inst}(O) = \mathcal{I}$
2. $\text{Sign}(O) = \Sigma$, where $(\mathcal{I}, \Sigma, \{\mu_i\}_{i \in |G|})$ is the colimit of the graph $G$ given by the semantics of $N$
3. $\text{Ax}(O) = \bigcup_{i \in |G|} \mu_i(\text{Ax}(O_i))$, where $O_i$ is the OMS label of the node $i$ in $G$
4. $\text{Mod}(O) = \{ M \in \text{Mod}(\Sigma) | M|_{\mu_i} \in \text{Mod}(O_i), i \in |G| \}$, where $O_i$ is the OMS label of the node $i$ in $G$.

3.2.11 MINIMISATION

A minimisation of an OMS imposes a closed-world assumption on part of the OMS. It forces the non-logical symbols declared in $O$ to be interpreted in a minimal way. This is written $\text{minimize } \{ O \}$. 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 \cite{30,3}. Alternatively, the non-logical symbols to be minimised and to be varied can be explicitly declared: $O \ \text{minimize } \Sigma_1 \ \text{vars } \Sigma_2$.

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

The semantics of $O = \text{minimize } O'$ is

- $\text{Inst}(O) = \text{Inst}(O')$
- $\text{Sign}(O) = \text{Sign}(O')$
- $\text{Mod}(O) = \{ M \in \text{Mod}(O') \mid M \text{ is minimal in } \text{Mod}(O') \}$
- $\text{Ax}(O)$ is undefined.

Note that for minimality we need the inclusions in model categories. Dually to minimisations, there are also maximisations.

3.2.12 HIDE VS. REMOVE VS. FORGET VS. REJECT

We have four ways of removing the class VegetarianTopping from the ontology Pizza using the keywords hide, remove, forget, and reject, respectively. Table 1 illustrates some of the connections between (3.2.6)–(3.2.9).

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 $O \text{ hide } \Sigma$ might itself not be a model of $O \text{ hide } \Sigma$. In examples involving “keep $L$”, 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 $L$. Also, an extracted module does not contain (*), because it only selects a subontology, and Pizza does not contain (*). Finally, Pizza reject VegetarianTopping simply drops all sentences involving VegetarianTopping, and therefore also consequences like (*) are lost.

Note that while forget/keep and hide/reveal both work w.r.t. smaller signatures and sublogics, remove/extract and select/reject do 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. In the case of select/reject, it is unclear what selecting of a sublogic should bring other than projecting to the sublogic using hide along.
Proposition 3.1 The following relations among the constructs in Table 1 hold:

\[
\begin{align*}
\text{Mod}(O \ hide \ \Sigma) &= \text{Mod}(O \ remove \ \Sigma) |_{\text{Sig}(O) \ \Sigma} \\
&\subseteq \text{Mod}(O \ forget \ \Sigma) \\
&\subseteq \text{Mod}(O \ reject \ \Sigma)
\end{align*}
\]

3.3 OMS Mappings

OMS mappings are generated by the following grammar:

\[
\text{MappingDefn ::= interpretation NAME : OMS to OMS = } \sigma \\
| \text{entailment } NAME = \text{OMS entails OMS} \\
| \text{equivalence } NAME : \text{OMS } \leftrightarrow \text{OMS } = (\Sigma, \Delta) \\
| \text{conservative extension } NAME = \text{O1 of O2 for } \Sigma \\
| \text{alignment } NAME \text{ CARD1 CARD2 : OMS to OMS } = \text{Correspondences} \\
| \text{refinement } NAME : \text{OMS to OMS } = \sigma \\
| \text{refinement } NAME = \text{NAME then NAME}
\]

The semantics of an OMS mapping is given as a graph whose nodes \( N \) are labeled with

- \( \text{Name}(N) \), the name of the node
- \( \text{Inst}(N) \), the institution of the node
- \( \text{Sign}(N) \), the signature of the node
- \( \text{Mod}(N) \), the class of \( \text{Sign}(N) \)-models of the node
- \( \text{Ax}(N) \), the set of \( \text{Ax}(N) \)-sentences of the node
and whose edges are labeled with signature morphisms between the signatures of the source and target nodes. The theory of a node corresponding to an elusive OMS may be undefined. The class of models of a node corresponding to a flattenable OMS is the class of models of $\text{Ax}(N)$. For brevity, we may write the label of a node as a tuple. We make the simplifying assumption that any OMS is assigned a unique name. The theory-level semantics of an OMS is needed for alignments.

In the following we discuss the different types of OMS mappings.

3.3.1 INTERPRETATION

Theory interpretations, written $\text{interpretation } Id : O_1 \to O_2 = \sigma$, 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:

$$\text{interpretation } i : \text{TotalOrder} \to \text{Nat} = \text{Elem} \mapsto \text{Nat}$$

Here is a more complex example in Common Logic from the COLORE repository [9]:

$$\text{interpretation } \text{geometry_of_time} \% \text{mcons} : \text{%% Interpretation of linearly ordered time intervals...}$$

$$\text{int:outtime_le}$$

$$\text{%% ... that begin and end with an instant as lines}$$

$$\text{%% that are incident with linearly ...}$$

$$\text{to \{ ord:linear_ordering and bi:complete_graphical}$$

$$\text{%% ... ordered points in a special geometry, ...}$$

$$\text{and int:mappings/outtime_interval_reduction }\}$$

$$= \text{int:ProperInterval} \mapsto \text{int:Interval} \text{end}$$

The semantics of $\text{interpretation } N : O_1 \to O_2 = \sigma$ is defined iff $\sigma$ is a signature morphism from $\text{Sign}(O_1)$ to $\text{Sign}(O_2)$ such that for each $M_2 \in \text{Mod}(O_2), M_2|\sigma \in \text{Mod}(O_1)$. In that case, the graph of $N$ is $\left(O_1, \text{Inst}(O_1), \text{Sign}(O_1), \text{Mod}(O_1), \text{Ax}(O_1)\right) \xrightarrow{\sigma} \left(O_2, \text{Inst}(O_2), \text{Sign}(O_2), \text{Mod}(O_2), \text{Ax}(O_2)\right)$

3.3.2 REFINEMENT

Refinements, written $\text{refinement } Id : O_1 \to O_2 = \sigma$, expressing that $O_2$ is an acceptable realisation of $O_1$. Semantically, this is equivalent with a theory interpretation from $O_1$ to $O_2$ along $\sigma$. Refinements can be combined using the $\text{then}$ keyword, as in the example below, where the requirement of implementing a monoid is refined to implementing the monoid of natural numbers with addition, using the representation of numbers as lists of binary digits, for efficiency:

$$\text{spec Monoid} =$$

$$\text{sort Elem}$$

$$\text{ops 0 : Elem;}$$

$$\quad \text{--+-- : Elem * Elem -> Elem, assoc, unit 0}$$

$$\text{end}$$

$$\text{spec NatWithSuc} =$$

$$\text{free type Nat ::= 0 | suc(Nat).}$$

$$\text{op __+__ : Nat * Nat -> Nat, unit 0}$$

$$\text{forall x , y : Nat . x + suc(y) = suc(x + y)}$$
op 1: Nat = suc(0)
end

spec Nat =
  NatWithSuc hide suc
end

refinement R1 =
  Monoid refined via Elem |-> Nat to Nat
end

spec NatBin =
generated type Bin ::= 0 | 1 | _-0(Bin) | _-1(Bin)
ops _+_, _++_ : Bin * Bin -> Bin
forall x, y : Bin.
  0 0 = 0 . 0 1 = 1 .
  not (0 = 1) . x 0 = y 0 => x = y
  not (x 0 = y 1) . x 1 = y 1 => x = y
  0 + 0 = 0 . 0 ++ 0 = 1
  x 0 + y 0 = (x + y) 0 . x 0 ++ y 0 = (x ++ y) 0
  x 1 + y 0 = (x + y) 1 . x 1 ++ y 0 = (x ++ y) 0
  x 1 + y 1 = (x + y) 1 . x 1 ++ y 1 = (x ++ y) 1
end

refinement R2 =
  Nat refined via Nat |-> Bin to NatBin
end

refinement R3 = R1 then R2

The semantics of refinement \( R : O_1 \to O_2 = \sigma \) is defined iff \( \sigma \) is a signature morphism from \( \text{Sign}(O_1) \) to \( \text{Sign}(O_2) \) such that for each \( M_2 \in \text{Mod}(O_2), M_2|_{\sigma} \in \text{Mod}(O_1) \). In that case, the graph of \( N \) is \( (O_1, \text{Inst}(O_1), \text{Sign}(O_1), \text{Mod}(O_1), \text{Ax}(O_1)) \sigma \to (O_2, \text{Inst}(O_2), \text{Sign}(O_2), \text{Mod}(O_2), \text{Ax}(O_2)) \).

The semantics of \( R_1 \) then \( R_2 \) is defined if and only if:
the semantics of \( R_1 \) is \( (N_1, I_1, \Sigma_1, M_1, \Delta_1) \xrightarrow{\sigma} (N_2, I_2, \Sigma_2, M_2, \Delta_2) \), and
the semantics of \( R_2 \) is \( (N'_1, I'_1, \Sigma'_1, M'_1, \Delta'_1) \xrightarrow{\sigma'} (N'_2, I'_2, \Sigma'_2, M'_2, \Delta'_2) \), such that \( I_2 = I'_1 \), \( \Sigma_2 = \Sigma'_1 \) and \( M'_1 \subseteq M'_2 \), and then the graph of the composition is \( (N''_1, I_1, \Sigma_1, \{M|_{\sigma_1, \sigma_2} | M \in \{M'_1\}_1, \bot\}) \sigma_1,\sigma_2 \to (N''_2, I_2, \Sigma_2, \{M''_2\}_2, \Delta''_2) \), where \( N''_1 \) is a new name.

3.3.3 Entailment

Entailments, written entailment \emph{Id} = \( O_1 \text{ entails } O_2 \), express that \( O_2 \) is logically entailed by \( O_1 \). For example, we can express that in a group, the inverse of an element still exists after hiding the explicit inverse operation from the specification as follows:

logic CASL

spec InterpolatedGroup =
sort Elem
mod Inst Mod equivalences OMS Boolean rings: O that classes that are in bijective correspondence. This is done by providing a (fragment) OMS case, the graph of N.
The semantics of equivalence Mod there exists a unique model M shall hold for any module Mod O then.

3.3.4 OMS Equivalence

OMS equivalences, written equivalence Id: O₁ ↔ O₂ = O₃, expressing that O₁ and O₂ have model classes that are in bijective correspondence. This is done by providing a (fragment) OMS O₃ such that O₁ then O₃ is a definitional extension [28]. For example, Boolean algebras are equivalent to Boolean rings:

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

The semantics of equivalence N : O₁ ↔ O₂ = O₃ is defined iff for each model Mᵢ ∈ Mod(Oᵢ) there exists a unique model M ∈ Mod(Sign(O₁), Sign(O₂), Σ(O₃)) such that Mᵢ|Sign(O₁) = Mᵢ. In that case, the graph of N is (O₁, Inst(O₁), Sign(O₁), Mod(O₁), Ax(O₁)) \(\cong\) (O₂, Inst(O₂), Sign(O₂), Mod(O₂), Ax(O₂)).

3.3.5 Conservative Extension

A conservative extension is written conservative extension Id c : O₁ of O₂ for Σ. This expresses that O₂ contains all knowledge about the signature Σ from the O₁ or, more precisely, O₁ is a conservative extension of O₂ with restriction signature Σ and conservativity c. If c is %scons, this means that every Σ-reduct of an O₂-model can be expanded to an O₁-model. If c is %ccons, this means that every Σ-sentence φ following from O₁ already follows from O₂. This relation shall hold for any module O₂ extracted from O₁ using the extract construct. For example, we
can specify that we obtained a module of GALEN by extracting the parts corresponding to drugs, joints and body parts as follows:

```plaintext
module myGalenIsAModule : myGalen of
  <http://example.org/GALEN/galen.owl>
  for Drugs, Joints, Bodyparts
end
```

The semantics of conservative extension $N c : O_1$ of $O_2$ for $\Sigma$ is defined iff $\Sigma \subseteq \text{Sign}(O_2) \subseteq \text{Sign}(O_1)$ and if $c = \% \text{mcons}$ and for each $M \in \text{Mod}(O_2)$ there is a model $M' \in \text{Mod}(O_1)$ such that $M'|\Sigma = M|\Sigma$, or if $c = \% \text{ccons}$ and for each $\varphi \in \text{Sen}(\Sigma)$, $O_1 \models \varphi$ implies $O_2 \models \varphi$. Then the graph of $N$ is $(O_2, \text{Inst}(O_2), \text{Sign}(O_2), \text{Mod}(O_2), \text{Ax}(O_2)) \xrightarrow{t} (O_1, \text{Inst}(O_1), \text{Sign}(O_1), \text{Mod}(O_1), \text{Ax}(O_1))$, with $t$ being the inclusion.

### 3.3.6 Alignment Definition

Alignment definitions, written alignment $Id$ card$_1$ card$_2 : O_1$ to $O_2 = c_1, \ldots, c_n$, assuming domain 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 $\text{sym}_1 \rel \text{conf} \text{sym}_2$. Here, $\text{sym}_i$ is a symbol from $O_i$, $\rel$ is one of the built-in relations $>$, $<$, $=$, $\%$, $\in$, $\mapsto$, or an identifier of a relation specified externally, and $\text{conf}$ is an (optional) confidence value between 0 and 1. The user can specify the assumption about the universe where the relations in the correspondences are interpreted using the assuming clause, with possible values SingleDomain (all ontologies are interpreted over the same universe, which is also the default), GlobalDomain (the domains of the ontologies are reconciled w.r.t. a global domain of interpretation) and ContextualizedDomain (the domains are connected via relations). This syntax of alignments follows the Alignment API [10].

If all correspondences of an alignment have the confidence value 1, the alignment can be given a formal semantics as a network.

#### Ontology

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

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

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

Note that BioPortal’s [42] 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.
network \( N = \)
\[ 1 : \text{Onto1}, \]
\[ 2 : \text{Onto2}, \]
\[ \text{VAlignment} \]

ontology \( \text{VAlignedOntology} = \)
\[ \text{combine } N \]
\[ \%\% 1: \text{Person} \text{ is identified with } 2: \text{HumanBeing} \]
\[ \%\% 1: \text{Woman} \text{ is identified with } 2: \text{Woman} \]
\[ \%\% 1: \text{Bank} \text{ and } 2: \text{Bank} \text{ are kept distinct} \]
end

ontology \( \text{VAlignedOntologyRenamed} = \)
\[ \text{VAlignedOntology with } 1: \text{Bank} \rightarrow \text{RiverBank}, 2: \text{Bank} \rightarrow \text{FinancialBank}, \]
\[ \text{Person_HumanBeing} \rightarrow \text{Person} \]
end

We sketch the semantics of alignments with the case when the domain of interpretation is assumed to be shared by the ontologies being aligned. In this case, the semantics is given by a \( W \)-shaped graph like in Fig. 2 where \( O_1 \) and \( O_2 \) are the nodes of the ontologies being aligned, \( O'_1 \) and \( O'_2 \) collect the symbols of \( O_1 \) and \( O_2 \), respectively, that appear in the correspondences of the alignment, \( \iota_1 \) and \( \iota_2 \) are inclusions and the bridge ontology \( B \) together with the morphisms \( \sigma_1 \) and \( \sigma_2 \) is constructed by turning the correspondences into bridge axioms. Details can be found in [8].

![Figure 2: Semantics of alignments](image)

### 3.4 Networks of OMS

OMS networks are introduced by the following grammar:

\[
\text{NetworkDefn} := \text{network NAME = Network} \\
\text{Network} := \text{NAME* [ excluding NAME* ]}
\]

Here, the \text{NAMEs} can name OMS, mappings or other networks. A network is specified as a list of network elements (OMS, OMS mappings and sub-networks), followed by an optional list of excluded network elements. For disambiguating the symbols in the combination of the network, the individual OMS can be prefixed with labels, like \( n: O \), which are scoped to the current OMS network. An example has already been presented in the section on alignments. Together with two OMS included in the network, the graph of the network implicitly includes all paths along importations between the two nodes. For example, in the example below, NAT_PLUS imports the
specification NAT. Without the implicit inclusion of this import, the combination would duplicate the theory of NAT.

```plaintext
spec NAT =
  free type Nat ::= 0 | suc(Nat)
end

spec NAT_PLUS =
  NAT
  then
  op __+_-- : Nat * Nat -> Nat
  ...
end
...

network N =
  NAT, NAT_PLUS,...
end

spec N_COMBINED =
  combine N
end
```

Formally, the graph of a network is constructed by taking the union of all graphs of its constituents, provided that we regard the semantics of OMS as a graph with one node and no edges, and removing from it all subparts specified in the excluding clause.

### 3.5 Libraries of OMS

Libraries start with the keyword `library` and the name of the library, followed by a `qualification` choosing the OMS language, logic, and/or serialisation. This is followed by a list of imports of other libraries, definitions of OMS, OMS mappings, networks of OMS, or other qualifications which change the current logic. Optionally, a prefix map placed at the beginning of a library may be used to abbreviate IRIs. A library can also be the inclusion of an OMS written in a language corresponding to some institution.

```plaintext
%prefix(  
  bfo: <https://bfo.googlecode.com/svn/releases/1.1.1/>  
)%

library Parthood

logic CommonLogic

ontology BFOWithAssociatedAxioms =
  bfo:bfo.owl with translation OWL22CommonLogic
  then
  (forall (x y) (if (snap:properTemporalPartOf x y)  
    (exists (z) (and (snap:properTemporalPartOf z y)  
      (not (exists (w)  
        (and (snap:temporalPartOf w x)  
          (snap:temporalPartOf w z)  
        ))))))
end
```
Figure 3: Overview of logics in Ontohub

Note that the prefixes declared in an imported library are available in the imported library, as illustrated in the example above with the prefix $\text{snap:}$.

This completes our overview of DOL. The full syntax and semantics of DOL will be available at wiki.ontohub.org and has been submitted to OMG for standardisation. The most recent version of the document is available at ontioip.org.

4 Tool Support for DOL

Currently, DOL is supported by two tools: Ontohub and the Heterogeneous Tool Set (HETS). Ontohub (see http://ontohub.org) is a web-based repository engine for OMS that are written either in DOL or in some specific OMS language.

Ontohub provides means for organising OMS into repositories. The distributed nature enables communities to share and exchange their contributions easily. The heterogeneous nature makes it possible to integrate OMS written in various OMS languages. Ontohub supports a wide range of DOL-conforming OMS languages building on DOL and also supports DOL’s interpretations, equivalences and alignments. Users of Ontohub can upload, browse, search and annotate OMS and OMS libraries in various languages via a web front end. Figure 3 shows an excerpt of the 25 logics currently available in Ontohub.

The parsing and inference back end is the Heterogeneous Tool Set (Hets [36, 41], available at hets.eu). Hets supports a large number of basic OMS languages and logics, as well as the DOL.

21Ontohub’s sources are freely available at https://github.com/ontohub/ontohub
The structural information extracted from DOL OMS by Hets is stored in the Ontohub database and exposed to human users via a web interface and to machine clients as linked data.

## 5 Conclusion and Future Work

Interoperability between systems as well as reusability, we argued in the introduction to this paper, are critical challenges.

We here proposed to address these challenges by introducing two abstractions: firstly, we introduced the notion of OMS, spanning formalised ontologies, models, and specifications; secondly, we introduced the DOL language, an abstraction in the sense that it provides a structuring, module, and mapping language independently of the particular logical formalism used.

The work presented here brings together previous work pursued in a number of communities, including in particular logical pluralism, modular ontologies, algebraic specification, and modelling of systems. It therefore combines many isolated logical modelling and specification solutions into one coherent framework with formal semantics.

A number of open problems and challenges, however, 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 monotonic logics?

---

22 Some (but only few) of DOL’s features are still being implemented at the time of the writing of this paper.

23 “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.
• What is a useful abstract notion of OMS 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 OMS languages and translations?

Despite these challenges, we hope that the development of DOL will have a profound impact on ontology engineering practices as well as on the way the modelling, ontology, and specification communities interact and how the systems they develop may interoperate. The impact on communities can already be seen e.g. by the use of Ontohub/DOL for the FOIS 2014 ontology competition. We have illustrated the benefits of DOL for a wide range of use cases; including for a framework of heterogeneous modelling in UML [21][22], in biomedical ontology [27], for the specification of blending diagrams in computational creativity [24], and for the heterogeneous modelling of musical harmonies [6][12].

We hope that the future will bring many more diverse and interesting use cases for the DOL language.

Acknowledgements

The development of DOL was supported by the German Research Foundation (DFG), Project I1-[OntoSpace] of the SFB/TR 8 “Spatial Cognition”.

The project COINVENT acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open Grant number: 611553.

The authors would like to thank the OntoIOp working group for their valuable input, particularly Michael Grüninger, Maria Keet, Christoph Lange, and Peter Yim.

We also want to thank Yazmin Angelica Ibañez, Thomas Schneider and Carsten Lutz for valuable input on interpolation and module extraction.

References


