Abstract. We present an overview of the landscape of ontology languages, mostly pertaining to the first-order paradigm. In particular, we present a uniform formalisation of these languages based on the institution theoretical framework, allowing a systematic treatment and analysis of the translational relationships between the various languages and a general analysis of properties of such translations. We also discuss the importance of language translation from the point of view of ontological modularity and logical pluralism, and for the borrowing of tools and reasoners between languages.

Keywords. Ontology languages, logic translations, institution theory

Introduction and Motivation

Ontologies are nowadays being employed in a large number of diverse information-rich domains. While the OWL standard has lead to an important unification of notation and semantics, still many distinct formalisms are used for writing ontologies. Some of these, as RDF, OBO and UML, can be seen more or less as fragments and notational variants of OWL, while others, like F-logic and Common Logic (CL), clearly go beyond the expressiveness of OWL.\(^2\) Moreover, not only the underlying logics are different, but also the modularity constructs.

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

In this work, we lay the foundation for a distributed ontology language (DOL), which will allow users to use their own preferred ontology formalism while becoming
interoperable with other formalisms (see [23] for further details). The DOL language is in particular intended to be at the core of a new ISO standardisation effort on ontology interoperability (proposed in ISO/TC 37/SC 3). At the heart of our approach is a graph of ontology languages and translations. This graph will enable users to

- relate ontologies that are written in different formalisms (e.g. prove that the OWL version of the foundational ontology DOLCE is logically entailed by the first-order version);
- re-use ontology modules even if they have been formulated in a different formalism;
- re-use ontology tools like theorem provers and module extractors along translations between formalisms.

The paper is organised as follows. In Section 1, we recall institutions, which formalise the notion of logical system, and in Section 2, we will cast many known ontology languages as institutions (largely following [24], but casting e.g. F-logic as an institution for the first time). This is then repeated for institution comorphisms (formalising the notion of logic translation) in Sections 3 and 4. The latter section contains the key contribution of this work: a graph of comorphisms among logics used for ontologies, together with their main properties.

1. Institutions: Formalising the Notion of Logical System

When relating different ontology formalisms, it is helpful to use a meta-notion that formalises the intuitive notion of logical system. Goguen and Burstall have introduced institutions [15] exactly for this purpose. We assume some acquaintance with the basic notions of category theory and refer to [1] or [26] for an introduction.

Definition 1. An institution 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 $\text{Sen} : \text{Sign} \to \text{Set}$\footnote{$\text{Set}$ is the category having all small sets as objects and functions as arrows.} 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 $\sigma(\phi)$,
- a functor $\text{Mod} : \text{Sign}^{\text{op}} \to \text{CAT}$\footnote{$\text{CAT}$ is the category of categories and functors. Strictly speaking, $\text{CAT}$ is not a category but only a so-called quasicategory, which is a category that lives in a higher set-theoretic universe.}$ 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 $\models_{\Sigma} \subseteq |\text{Mod}(\Sigma)| \times \text{Sen}(\Sigma)$ for each $\Sigma \in |\text{Sign}|$,

such that for each $\sigma : \Sigma \to \Sigma'$ in $\text{Sign}$ the following satisfaction condition holds:

\begin{equation}
\tag{*} M' \models_{\Sigma'} \sigma(\varphi) \iff M'|_{\sigma} \models_{\Sigma} \varphi
\end{equation}
for each $M' \in |\text{Mod}(\Sigma')|$ and $\varphi \in \text{Sen}(\Sigma)$, expressing that truth is invariant under change of notation and context.5

A theory in an institution is a pair $\text{Th} = (\Sigma, \Gamma)$ consisting of a signature $\Sigma$ and a set $\Gamma$ of sentences over $\Sigma$. The models of $\text{Th}$ are those $\Sigma$-models that satisfy $\Gamma$. Satisfiability and logical consequence are defined in the standard way. Moreover, the following kernel language of modular specifications [32] can be interpreted in any institution:

$\text{SP} ::= (\Sigma, \Gamma) \mid \text{SP}_1 \cup \text{SP}_2 \mid \sigma(\text{SP}) \mid \sigma^{-1}(\text{SP})$

with the following semantics:

- $\text{Mod}((\Sigma, \Gamma)) = \{ M \in \text{Mod}(\Sigma) \mid M \models \Gamma \}$
- $\text{Mod}(\text{SP}_1 \cup \text{SP}_2) = \text{Mod}(\text{SP}_1) \cap \text{Mod}(\text{SP}_2)$
- $\text{Mod}(\sigma(\text{SP})) = \{ M \mid M|\sigma \in \text{Mod}(\text{SP}) \}$
- $\text{Mod}(\sigma^{-1}(\text{SP})) = \{ M|\sigma \mid M \in \text{Mod}(\text{SP}) \}$

Most modularity concepts used for ontologies can be mapped into this kernel language.

2. Ontology Languages as Institutions

We now cast a rather comprehensive list of well-known ontology languages as institutions, largely following [24], but also extending the list of formalisms given there by including F-logic, OBO, RDFS, and some modular ontology languages, but leaving out some of the less often used formalisms, such as fuzzy and paraconsistent DL.

**Definition 2** (Propositional Logic). The institution $\text{PL}$ of propositional logic has sets $\Sigma$ (propositional symbols) as signatures, and functions $\sigma : \Sigma_1 \rightarrow \Sigma_2$ between such sets as signature morphisms. A $\Sigma$-model $M$ is a mapping from $\Sigma$ to $\{\text{true}, \text{false}\}$. The reduct of a $\Sigma_2$-model $M_2$ along $\sigma : \Sigma_1 \rightarrow \Sigma_2$ is the $\Sigma_1$-model given by the composition $M_2 \circ \sigma$. $\Sigma$-sentences are built from $\Sigma$ with the usual propositional connectives, and sentence translation along a signature morphism just replaces the propositional symbols along the morphism. Finally, satisfaction of a sentence in a model is defined by the standard truth-table semantics. It is straightforward to see that the satisfaction condition holds. ⊣

Propositional reasoning is at the core of ontology design. Boolean expressivity is sufficient to axiomatise the taxonomic structure of an ontology by imposing disjointness and sub- or super-concept relationships via implication and negation, as well as e.g. non-empty overlap of concepts.

**Definition 3** (Untyped First-order Logic). In the institution $\text{FOL}^=_{\text{ut}}$ of untyped first-order logic with equality, signatures are first-order signatures, consisting of a set of function symbols with arities, and a set of predicate symbols with arities. Signature morphisms map symbols such that arities are preserved. Models are first-order structures, and sentences are first-order formulas. Sentence translation means replacement of the translated

---

5Note, however, that non-monotonic formalisms can only indirectly be covered this way, but compare, e.g., [18].
symbols. Model reduct means reassembling the model’s components according to the signature morphism. Satisfaction is the usual satisfaction of a first-order sentence in a first-order structure.

**Definition 4** (Many-sorted First-order Logic). The institution $\text{FOL}^{ms}$ of many-sorted first-order logic with equality is similar to $\text{FOL}^=$, the main difference being that signatures are many-sorted first-order signatures, consisting of sorts and typed function and predicate symbols, and that formulas need to be well-typed. For details, see [15].

Although not strictly more expressive than untyped $\text{FOL}^=$, introducing a sort structure allows a cleaner and more principled design of first-order ontologies. Moreover, axioms involving different sorts can be stated more succinctly, and static type checking gives more control over correct modelling.

**Definition 5** (Common Logic - CL). Common Logic (CL) has first been formalised as an institution in [24]. A CL signature $\Sigma$ (called vocabulary in CL terminology) consists of a set of names, with a subset called the set of discourse names, and a set of sequence markers. A signature morphism consists of two maps between these sets, such that the property of being a discourse name is preserved and reflected.

A $\Sigma$-model consists of a set $UR$, the universe of reference, with a non-empty subset $UD \subseteq UR$, the universe of discourse, and four mappings:

- $\text{rel}$ from $UR$ to subsets of $UD^* = \{ < x_1, \ldots, x_n > | x_1, \ldots, x_n \in UD \}$ (i.e., the set of finite sequences of elements of $UD$);
- $\text{fun}$ from $UR$ to total functions from $UD^*$ into $UD$;
- $\text{int}$ from names in $\Sigma$ to $UR$, such that $\text{int}(v)$ is in $UD$ if and only if $v$ is a discourse name;
- $\text{seq}$ from sequence markers in $\Sigma$ to $UD^*$.

Model reducts leave $UR$, $UD$, $\text{rel}$ and $\text{fun}$ untouched, while $\text{int}$ and $\text{seq}$ are composed with the appropriate signature morphism component. A $\Sigma$-sentence is a first-order sentence, where predications and function applications are written in a higher-order like syntax as $t(s)$. Here, $t$ is an arbitrary term, and $s$ is a sequence term, which can be a sequence of terms $t_1, \ldots, t_n$, or a sequence marker. However, a predication $t(s)$ is interpreted like the first-order formula $\text{holds}(t, s)$, and a function application $t(s)$ like the first-order term $\text{app}(t, s)$, where $\text{holds}$ and $\text{app}$ are fictitious symbols denoting the semantic objects $\text{rel}$ and $\text{fun}$. In this way, CL provides a first-order simulation of a higher-order language. Quantification variables are partitioned into those for individuals and those for sequences. Sentence translation along signature morphisms is done by simple replacement of names and sequence markers. Interpretation of terms and formulae is as in first-order logic, with the difference that the terms at predicate resp. function symbol positions are interpreted with $\text{rel}$ resp. $\text{fun}$ in order to obtain the predicate resp. function, as discussed above. A further difference is the presence of sequence terms (namely sequence markers and juxtapositions of terms), which denote sequences in $UD^*$, with term juxtaposition interpreted by sequence concatenation. Note that sequences are essentially a second-order feature. For details, see [11]. As an example, consider the DOLCE formula $\forall \phi(\phi(x))$, corresponding to $\bigwedge_{\psi \in \Pi} (\psi(x))$, where predicate variables $\phi, \psi$ range

---

That is, a name is a discourse name if and only if its image under the signature morphism is.
over a finite set \( \Pi \) of explicitly introduced universals. In CL, this is written, using standard logical syntax (note that CL is agnostic about concrete syntax)

\[
\forall \phi. \Pi(\phi) \rightarrow \phi(x)
\]

or in the often used Lisp-like syntax of the CL Interchange Format CLIF:

\[
(forall (?phi) (if (pi ?phi) (?phi ?x)))
\]

Sequence markers add even more flexibility. For example, it is possible to express that a list of predicates is mutually disjoint as follows (using the sequence marker "..."):

\[
mutually-disjoint(P) \\
\text{mutually-disjoint}(P Q \ldots) \iff (\forall x. \neg(P(x) \land Q(x))) \land \text{mutually-disjoint}(P \ldots) \land \text{mutually-disjoint}(Q \ldots)
\]

For the rationale and methodology of CL and the possibility to define dialects covering different first-order languages, see [11].

**Definition 6** (CASL). CASL (the Common Algebraic Specification Language, [4, 30]) provides an extension of many-sorted first-order logic with partial functions, subsorting and so-called sort-generation constraints. While partial functions and subsorting do not essentially add expressivity (they can be coded out), sort-generation constraints do: they are many-sorted induction axioms (of a second-order nature) that can be used for the definition of datatypes like natural numbers, lists, trees etc.

**Definition 7** (Relational Schemes - Rel-S). This logic, first introduced in [21], is about schemes for relational databases and their integrity constraints. A signature in this institution consists of a set of sorts and a set of relation symbols, where each relation symbol is indexed with a string of sorted field names as in:

- **paper**:
  
  \(\text{id} : \text{integer}, \text{title} : \text{string}, \text{published in} : \text{integer}\)

- **journal**:
  
  \(\text{id} : \text{integer}, \text{name} : \text{string}, \text{impact factor} : \text{float}\)

Some sorts for the relational schema as **integer**, **float** and **string** are predefined and equipped with default interpretations. The identifier **key** can be used as a prefix to sorted field names to specify the primary (compound) key of the schema. Signature morphisms map sorts, relation symbols and field names in a compatible way, such that primary keys are preserved. A model consists of a carrier set for each sort, where some sorts have predefined carrier sets, and an \(n\)-ary relation for each relation symbol with \(n\) fields. Model reduction is like that of many-sorted first-order logic. A sentence is a link (integrity constraint) between two field names of two relation symbols. For example, the link

\[
\text{paper[published in]} \rightarrow \text{journal[id]} \text{ one to many}
\]

requires that the field `published in` of any paper coincides with the `id` of at least one journal (the many-one character of this relationship is expressed by the keyword
one to many). Other possible relationships are one to one and many to many. Sentence translation is just renaming of relation symbols and of sorts. A link $r[f] \rightarrow s[g]$ is satisfied in case of $t = \text{one to many}$ if for each element in $r[f]$ there are zero or more occurrences of this element in $s[g]$, but for each element in $s[g]$ there is at most one occurrence of an element in $r[f]$. For $t = \text{one to one}$ in both cases only one occurrence is allowed, and for many to many there is no restriction on the number of occurrences.

**Definition 8** (Description Logics: OWL and its profiles EL, QL, RL). Signatures of the description logic $\mathcal{ALC}$ consist of a set $\mathcal{A}$ of atomic concepts, a set $\mathcal{R}$ of roles and a set $\mathcal{I}$ of individual constants, while signature morphisms provide respective mappings. Models are single-sorted first-order structures that interpret concepts as unary and roles as binary predicates. Sentences are subsumption relations $C_1 \sqsubseteq C_2$ between concepts, where concepts follow the grammar

$$C ::= A \mid \top \mid \bot \mid C_1 \sqcup C_2 \mid C_1 \sqcap C_2 \mid \neg C \mid \forall R.C \mid \exists R.C$$

These kind of sentences are also called TBox sentences. Sentences can also be ABox sentences, which are membership assertions of individuals in concepts (written $a : C$ for $a \in \mathcal{I}$) or pairs of individuals in roles (written $R(a, b)$ for $a, b \in \mathcal{I}, R \in \mathcal{R}$). Sentence translation and reduct is defined similarly as in FOL\textsuperscript{7}. Satisfaction is the standard satisfaction of description logics.

The logic $\text{SROIQ}$\textsuperscript{7} [19], which is the logical core of the Web Ontology Language OWL 2 DL\textsuperscript{7} extends $\mathcal{ALC}$ with the following constructs: (i) complex role boxes (denoted by $\mathcal{SR}$): these can contain: complex role inclusions such as $R \circ S \subseteq S$ as well as simple role hierarchies such as $R \subseteq S$, assertions for symmetric, transitive, reflexive, asymmetric and disjoint roles (called RBox sentences), as well as the construct $\exists R.\text{Self}$ (collecting the set of ‘$R$-reflexive points’); (ii) nominals (denoted by $\mathcal{O}$); (iii) inverse roles (denoted by $\mathcal{I}$); qualified and unqualified number restrictions ($\mathcal{Q}$). For details on the rather complex grammatical restrictions for $\text{SROIQ}$ (e.g. regular role inclusions, simple roles) compare [19], and see the example given below. $\text{SROIQ}$ can be straightforwardly rendered as an institutions following the previous examples, but compare also [25].

The OWL 2 specification contains three further DL fragments of $\text{SROIQ}$, called profiles, namely EL, QL, and RL\textsuperscript{8}. These are obtained by imposing syntactic restrictions on the language constructs and their usage, with the motivation that these fragments are of lower expressivity and support specific computational tasks. For instance, RL is designed to make it possible to implement reasoning systems using rule-based reasoning engines, QL to support query answering over large amounts of data, and EL is a sub-Boolean fragment sufficiently expressive e.g. for dealing with very large biomedical ontologies such as the NCI thesaurus. To sketch one of these profiles in some more detail, the (sub-Boolean) description logic $\mathcal{EL}$ underlying EL has the same sentences as $\mathcal{ALC}$ but restricts the concept language of $\mathcal{ALC}$ as follows:

$$C ::= B \mid \top \mid C_1 \sqcap C_2 \mid \exists R.C$$

Given that EL, QL, and RL are obtained via syntactic restrictions but leaving the overall $\text{SROIQ}$ semantics intact, it is obvious that they are subinstitutions of $\text{SROIQ}$. \textsuperscript{7}

\textsuperscript{8}See also http://www.w3.org/TR/owl2-overview/

\textsuperscript{8}See http://www.w3.org/TR/owl2-profiles/ for details of the specifications.
Apart from some exceptions\(^9\), description logics can be seen as fragments of first-order logic via the standard translation [2] that translates both the syntax and semantics of various DLs into untyped first-order logic. A similar situation obtains in the case of the OBO language designed for biomedical ontologies:

**Definition 9** (OBO). OBO is a very popular ontology language in the area of biomedical ontology engineering. On the syntactic side, it is straightforward to describe the language’s signatures and sentences in an institution theoretic style, but we here have to leave out the details of such a description. On the semantic side, OBO is a curious case of a language that has been used extensively and for which editors and even reasoners have been successfully implemented, relying initially on only informally specified semantics.

Beginning with OBO version 1.2, and building on an agreement with the OBO community concerning the informal semantics, it was realised that formal semantics could be (mostly\(^10\)) borrowed from OWL, see [17]. In the most recent version of OBO, version 1.4, the translation to OWL 2 as providing the formal semantics is now an official part of the draft specification.\(^11\) This is an instance of borrowing model theory in the sense of [9], by which an institution \(\text{OBO}^\text{OWL}\) is obtained. Since the translation is still partial, \(\text{OBO}^\text{OWL}\) is only a subset of the full OBO 1.4. For instance, in order to preserve decidability, \(\text{SROIQ}\) prohibits cardinality constraints on transitive object properties, whilst the full OBO 1.4 allows this. To render the full OBO 1.4 as an institution, the translation that defines the satisfaction relation between OBO sentences and the derived semantics has to be extended beyond \(\text{SROIQ}\). This is straightforward: the added constructs in OBO such as Boolean constructors on roles have a clear correspondent in DL semantics, which makes it straightforward to complete the mapping of the semantics.

\(^9\)For instance, adding transitive closure of roles or fixpoints to DLs makes them decidable fragments of second-order logic [5].

\(^10\)Some language constructs, such as ‘being necessarily false’ were seen to not have sufficiently clear semantics, and were subsequently dropped from the OBO language.

\(^11\)http://www.geneontology.org/GO.format.obo-1_4.shtml#OWL

---

**Definition 10** (RDF and RDFS). Following [25], we define the institutions for the Resource Description Framework (RDF) and RDF-Schema (RDFS), respectively. These are based on a logic called bare RDF (bRDF), which consists of triples only (without any predefined resources).

A signature \(R_s\) in bRDF is a set of resource references. For \(sub, pred, obj \in R_s\), a triple of the form \((sub, pred, obj)\) is a sentence in bRDF, where \(sub, pred, obj\) represent subject name, predicate name, object name, respectively. An \(R_s\)-model \(M = \langle R_m, P_m, S_m, \text{EXT}_m \rangle\) consists of a set \(R_m\) of resources, a set \(P_m \subseteq R_m\) of predicates, a mapping function \(S_m : R_s \rightarrow R_m\), and an extension function \(\text{EXT}_m : P_m \rightarrow \mathcal{P}(R_m \times R_m)\) mapping every predicate to a set of pairs of resources. Satisfaction is defined as follows:

\[
\mathcal{M} \models_{R_s} (sub, pred, obj) \iff (S_m(sub), (S_m(obj)) \in \text{EXT}_m(S_m(pred)).
\]

Both RDF and RDFS are built on top of bRDF by fixing a certain standard vocabulary both as part of each signature and in the models. Actually, the standard vocabulary is given by a certain theory. In case of RDF, it contains e.g. resources \text{rdf:type}
and \texttt{rdf:Property} and \texttt{rdf:subject}, and sentences (\texttt{rdf:type, rdf:type, rdf:Property}), and (\texttt{rdf:subject, rdf:type, rdf:Property}).

In the models, the standard vocabulary is interpreted with a fixed model. Moreover, for each RDF-model \( M = (R_m, P_m, S_m, EXT_m) \), if \( p \in P_m \), then it must hold \((p, S_m(\texttt{rdf:Property})) \in EXT_m(\texttt{rdf:type})\). For RDFS, similar conditions are formulated (here, for example also the subclass relation is fixed).

In the case of RDFS, the standard vocabulary contains more elements, like \texttt{rdf:domain, rdf:range, rdf:Resource, rdf:Literal, rdf:Datatype, rdf:Class, rdf:subClassOf, rdf:subPropertyOf, rdf:member, rdf:Container, rdf:ContainerMembershipProperty}.

There is also RDFS\textsuperscript{OWL}, an extension of RDFS with resources like \texttt{owl:Thing} and \texttt{owl:oneOf}, tailored towards the representation of OWL.

**Definition 11** (Modular Ontology Languages: \(\mathcal{E}\)-connections and DDL). \(\mathcal{E}\)-connections can be considered as many-sorted heterogeneous theories: component ontologies can be formulated in different logics, but have to be built from many-sorted vocabulary, and link relations are interpreted as relations connecting the sorts of the component logics.

The main difference between distributed description logics (DDLs) \cite{DDL} and various \(\mathcal{E}\)-connections now lies in the expressivity of the ‘link language’ \(\mathcal{L}\) connecting the different ontologies. While the basic link language of DDL is a certain sub-Boolean fragment of many sorted \(ALC\), the basic link language of \(\mathcal{E}\)-connections is \(ALC\textsuperscript{ms}\).\footnote{More precisely, \(ALC\textsuperscript{ms}\) here comprises existential and universal restrictions on link relations (and their inverses) with Booleans belonging to the components. This can be weakened to e.g. sub-Boolean DL, or strengthened to more expressive many-sorted DLs involving e.g. number restrictions or Boolean operators on links, see \cite{ECo} for details.}

The idea to ‘connect’ logics can be elegantly generalised to the institutional level (compare \cite{BaaderGhilardi} who note that their ‘connections’ are an instance of a more general comma construction). Without giving the full details of such a generalisation, it should be clear that, intuitively, we need to formalise the idea that an abstract connection of two logics \(S_1\) and \(S_2\) is obtained by defining a bridge language \(\mathcal{L}(\mathcal{E})\), where the elements of \(\mathcal{E}\) go across the sort-structure of the respective logics, and where theory extensions (containing the bridge axioms) are defined over a new language defined from the disjoint union of the original languages together with \(\mathcal{L}(\mathcal{E})\), containing certain expressive means applied (inductively) to the vocabulary of \(\mathcal{E}\).

Note that this generalises the \(\mathcal{E}\)-connections of \cite{ECo}, the DDLs of \cite{DDL}, as well as the connections of Baader and Ghilardi \cite{BaaderGhilardi} in two important respects: first, the institutional level generalises the term-based abstract description languages (ADS) that are an abstraction of modal and description logics, and second, the rather general definition of bridge theory similarly abstracts from the languages previously employed for linking that were similarly inspired by modal logic operators.

Given this, the phrasing of DDL and \(\mathcal{E}\)-connections as institutions is easily obtained from the component institutions: the institution of DDL\textsuperscript{OWL} is the institution whose component logics are OWL based and whose bridge rules follow DDL restrictions, the institution of \(\mathcal{E}\text{Co}^{\text{OWL}}\) allows only OWL-based components, but allows the more general bridge expressivity of \(\mathcal{E}\)-connections,\footnote{Note that allowing full OWL expressivity on the link language leads to undecidability also for OWL-based components.} and \(\mathcal{E}\text{Co}^{\text{FOL}}\) is the institution whose components can be build from full first-order logic, and whose bridge rules allow full first-order logic over link relations.
It should then be rather clear that e.g. $E$-connections of OWL ontologies can be encoded within ‘many-sorted’ $SROIQ$, with additional syntactic restrictions capturing the allowed bridge axioms, see also Section 4.

**Definition 12** (F-Logic). F-logic [20] is an object-oriented extension of first-order logic. For simplicity, we here treat the monotonic part of F-logic only, since this is a logic in the classical sense and can hence be formalised as an institution. The non-monotonic part should be formalised with methods like logic programming over an institution, see [13].

F-logic inherits signatures from $FOL^+$. Sentences are first-order sentences with equality, with the following additional formulas:

- **is-a assertions** $O : C$ expressing membership of an object in a class,
- **subclass assertions** $C :: D$,
- **object atoms of the form** $O[me]$, where the $me$ is a method expression\(^{14}\).

Method expression have the following forms:

- **non-inheritable scalar expressions** $ScalarMethod @ t_1, \ldots, t_n \rightarrow t$,
- **non-inheritable set-valued expressions** $SetMethod @ t_1, \ldots, t_n \rightarrow \{u_1, \ldots, u_m\}$,
- **inheritable scalar expressions** $ScalarMethod @ t_1, \ldots, t_n \bullet \rightarrow t$,
- **inheritable set-valued expressions** $SetMethod @ t_1, \ldots, t_n \bullet \rightarrow \{u_1, \ldots, u_m\}$,
- **scalar signature expressions** $ScalarMethod @ t_1, \ldots, t_n \Rightarrow (u_1, \ldots, u_m)$,
- **set-valued signature expressions** $SetMethod @ t_1, \ldots, t_n \Rightarrow (u_1, \ldots, u_m)$.

Here, $ScalarMethod$, $SetMethod$ and the $t_i$ and $u_i$ are terms.

Models are first-order structures (unsorted, that is, over a universe $U$) equipped with additional components serving for the interpretation of the additional syntax:

- $:$ is interpreted with a binary relation $\varepsilon$, and $::$ with an irreflexive partial order $\prec$.
- for $u \in U$ and each $n \geq 0$, there are
  - partial functions $P^n_\varepsilon (u), P^n_\prec (u) : U^{n+1} \rightarrow U$,
  - partial functions $I^n_\varepsilon (u), I^n_\prec (u) : U^{n+1} \rightarrow \mathcal{P}(U)$,
  - partial anti-monotonic functions $I^n_\varepsilon (u), I^n_\prec (u) : U^{n+1} \rightarrow \mathcal{P}_1(U)$, where $\mathcal{P}_1(U)$ is the set of upward-closed (w.r.t. $\prec$) subsets of $U$.

Satisfaction is defined like for first-order logic, where $:$ and $::$ are interpreted with $\varepsilon$ and $\prec$, respectively. An object atom $O[ScalarMethod @ t_1, \ldots, t_n \rightarrow t]$ holds in a model $M$ under a variable valuation $\nu$, if $I^n_{\varepsilon}(\nu(ScalarMethod))(\nu(O), \nu(t_1), \ldots, \nu(t_n))$ is defined and equal to $\nu(t)$; similarly for $::$. $O[SetMethod @ t_1, \ldots, t_n \rightarrow \{u_1, \ldots, u_m\}]$ holds in $M$ w.r.t. $\nu$, if $I^n_{\prec}(\nu(SetMethod))(\nu(O), \nu(t_1), \ldots, \nu(t_n))$ is defined and contains the set $\{\nu(u_1), \ldots, \nu(u_m)\}$; similarly for $::$, $\Rightarrow$ and $\Leftrightarrow$.

Having so many different arrows with the same semantics seems superfluous at first sight; their use will become clear when looking at type-checking and non-monotonic inference, which are defined on top of the logic given here. For this, as well as the rationale and the methodology for the use of F-logic in the field of object-oriented modelling, see [20].

\(^{14}\)Object molecules $O[me_1; \ldots; me_n]$ abbreviate conjunctions of object atoms.
Definition 13 (HOL). [7] presents an institution for a higher-order logic extending Church’s type theory [10] with polymorphism; this is basically the higher-order logic used in modern interactive theorem provers like Isabelle/HOL [31] (one additional feature of Isabelle are type classes).

3. Institution Comorphisms: Formalising Logic Translations

We will formalise ontology languages (logics) as institutions and ontology language translations as so-called institution comorphisms, see [16, 29]:

Definition 14 (Institution Comorphism). Given two institutions $I$ and $J$ with $I = (\text{Sign}^I, \text{Mod}^I, \text{Sen}^I, \models^I)$ and $J = (\text{Sign}^J, \text{Mod}^J, \text{Sen}^J, \models^J)$, an institution comorphism from $I$ to $J$ consists of a functor $\Phi : \text{Sign}^I \rightarrow \text{Sign}^J$, and natural transformations $\beta : \text{Mod}^J \circ \Phi = \Rightarrow \text{Mod}^I$ and $\alpha : \text{Sen}^I = \Rightarrow \text{Sen}^J \circ \Phi$, such that

\[ M' \models^J \Phi(\Sigma) \alpha_{\Sigma}(\varphi) \Leftrightarrow \beta_{\Sigma}(M') \models^I \Sigma \varphi. \]

holds, called the satisfaction condition.

Here, $\Phi(\Sigma)$ is the translation of 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.

A simple theoroidal comorphism is like a comorphism, except that the signature translation functor $\Phi$ maps to the category of theories over the target institution.

A simple example is given by considering the well-known translation of OWL into untyped first-order logic, mapping concepts to unary and roles to binary predicates. We will give the details of this paradigmatic case in Section 4.

The practical usefulness of institution comorphisms grows with their properties. An important property is model expansion, which is similar to conservative extension and ensures that logical consequence is represented faithfully. The amalgamation property ensures good interaction with modular specifications. Finally, subinstitution comorphisms capture the notion of sublogic.

Definition 15. An institution comorphism is model-expansive, if each model translation $\beta_{\Sigma}$ is surjective on objects.

Let $\rho = (\Phi, \alpha, \beta) : I \rightarrow J$ be an institution comorphism and let $D$ be a class of signature morphisms in $I$. Then $\rho$ is said to have the (weak) $D$-amalgamation property, if for each signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2 \in D$, the diagram

\[
\begin{array}{ccc}
\text{Mod}^I(\Sigma_2) & \overset{\beta_{\Sigma_2}}{\leftarrow} & \text{Mod}^I(\Phi(\Sigma_2)) \\
\text{Mod}^I(\sigma) & \downarrow & \text{Mod}^I(\Phi(\sigma)) \\
\text{Mod}^I(\Sigma_1) & \overset{\beta_{\Sigma_1}}{\leftarrow} & \text{Mod}^I(\Phi(\Sigma_1))
\end{array}
\]
admits (weak) amalgamation, i.e. any for any two models \( M_2 \in \text{Mod}^I(\Sigma_2) \) and \( M'_1 \in \text{Mod}^J(\Phi(\Sigma_1)) \) with \( M_2|_{\sigma} = \beta_{\Sigma_1}(M'_1) \), there is a unique (not necessarily unique) \( M'_2 \in \text{Mod}^I(\Phi(\Sigma_2)) \) with \( \beta_{\Sigma_2}(M'_2) = M_2 \) and \( M'_2|_{\Phi(\sigma)} = M'_1 \). In case that \( D \) consists of all signature morphisms, the (weak) \( D \)-amalgamation property is also called (weak) \( D \)-exactness. If we omit \( D \), we understand it to consist of all monomorphisms (typically, these are the injective morphisms).

An institution comorphism \( \rho = (\Phi, \alpha, \beta): I \to J \) is said to be model-isomorphic if for each \( \Sigma \in \text{Sign}^I \), \( \beta_{\Sigma} \) is an isomorphism. It is a subinstitution comorphism [27], if moreover \( \Phi \) is an embedding and each \( \alpha_{\Sigma} \) is injective. The intuition is that theories should be embedded, while models should be represented exactly (such that modal-theoretic results carry over).

It is easy to see that a model-isomorphic comorphism also is model-expansive and exact. Examples will be given in Section 4.

The crucial results for comorphisms with good properties are “borrowing” results, that is, a proof calculus or theorem prover capturing logical consequence in the target logic of the comorphism can be borrowed for deciding logical consequence also in the source logic.

**Proposition 16 (Borrowing [9]).** Let \( \rho = (\Phi, \alpha, \beta): I \to J \) be an institution comorphism, \( \Sigma \) a signature in \( I \) and \( \Gamma \cup \{\varphi\} \) a set of \( \Sigma \)-sentences. Then

\[
\Gamma \models^I \varphi \implies \alpha_{\Sigma}(\Gamma) \models^J \alpha_{\Sigma}(\varphi),
\]

\[
\text{\( \Gamma \) satisfiable} \iff \alpha_{\Sigma}(\Gamma) \text{ satisfiable},
\]

and if \( \rho \) is model-expansive, also the converse directions hold. Moreover, if \( SP \) is a modular specification and \( \rho \) is exact, then [8]

\[
SP \models^I \varphi \iff \rho(SP) \models^J \alpha_{\Sigma}(\varphi),
\]

\[
SP \text{ satisfiable} \iff \rho(SP) \text{ satisfiable},
\]

where \( \rho(SP) \) is the translation of \( SP \) using \( \Phi \) and \( \alpha \).

4. The Onto-Logical Translation Graph

Little work has been devoted to the general problem of translation between ontologies formulated in different logical languages and/or vocabularies. One such approach is given in [14], who discuss translations between OWL ontologies. They use so-called bridging axioms (formulated in first-order) to relate the meaning of terms in different ontologies,\(^\text{15}\) and present an algorithm to find such translations. More prominent in the ontology engineering world are of course the standard translation into first-order logic,

\(^{15}\)Not to be confused with the ‘bridge axioms’ in DDL [6].
which essentially ‘coincides’ with the direct semantics of OWL, and more interestingly
the case of OBO discussed above, where the logic translation delivers a definition of
formal semantics for the OBO language (which it itself does not have).

We here present an overview of logic-translations between the common ontology
languages as introduced above. Note that the resulting graph of logics and translations
can be used in several ways: one way is to take some logic high up in the graph, like HOL,
and map every ontology into it. While this universal, all-purpose approach may make
sense from a semantic point of view, it makes little sense from a practical point of view:
a more distributed, multilateral and pluralist approach has the advantage that specialised
tools can be used, whilst still interfacing ontologies written in different languages.

Substitutions: EL $\rightarrow$ OWL, QL $\rightarrow$ OWL and RL $\rightarrow$ OWL and FOL $\equiv$ $\rightarrow$ FOL$^{\text{ms}}$ $\equiv$
are obvious substitutions.

OWL $\rightarrow$ FOL$^{\equiv}$ is a straight-forward extension of the standard translation [5]
mapping individuals to constants, classes to unary predicates and roles to
binary predicates.

$\mathcal{E}Co_{\text{OWL}}$ $\rightarrow$ $\mathcal{E}Co_{\text{FOL}^{\equiv}}$ uses OWL $\rightarrow$ FOL$^{\equiv}$ twice, at the level of the base logic
and at the level of the bridge rules.

PL $\rightarrow$ FOL$^{\text{ms}}$ is a subinstitution by mapping propositional variables to nullary
predicates.
OBO\textsuperscript{OWL} \rightarrow \textit{OWL}: signatures and sentences are translated according to the OBO standard, whereas the model translation is the identity (due to borrowing of model theory).

\textit{OBO1.4} \rightarrow \textit{FOL}\textsuperscript{=} extends the composition \textit{OBO}\textsuperscript{OWL} \rightarrow \textit{OWL} \rightarrow \textit{FOL}\textsuperscript{=} by an explicit straight-forward coding of the additional features not present in \textit{OBO}\textsuperscript{OWL}.

\textit{bRDF} \rightarrow \textit{FOL}\textsuperscript{=} The subinstitution comorphism from \textit{bRDF} to \textit{FOL}\textsuperscript{=} maps a \textit{bRDF} signature \textit{R}_a to the \textit{FOL}\textsuperscript{=} signature \textit{Φ(}\textit{R}_a\textit{)} which has \textit{R}_a as set of constants, and moreover is equipped with a unary predicate \textit{P} and a ternary predicate \textit{EXT}. A \textit{bRDF}-sentence \textit{(sub, pred, obj)} is translated to \textit{EXT(}\textit{sub, pred, obj}\textit{)}. Finally, a \textit{FOL}\textsuperscript{=}-model of \textit{Φ(}\textit{R}_a\textit{)} is translated to the \textit{bRDF} which has the model’s universe as set of resources \textit{R}_m, while \textit{P}_m is given by the interpretation of \textit{P} and \textit{S}_m by the interpretation of the constants. \textit{EXT}_m can be easily constructed from the interpretation of \textit{EXT}.

\textit{FOL}\textsuperscript{=} \rightarrow \textit{F}-logic is an obvious subinstitution.

\textit{bRDF} \rightarrow \textit{RDF}: this is an obvious inclusion, except that \textit{bRDF} resources need to be renamed if they happen to have a predefined meaning in \textit{RDF}. The model translation needs to forget the fixed parts of RDF models, since this part can always be reconstructed in a unique way, we get an isomorphic model translation. \textit{RDF} \rightarrow \textit{RDFS} and \textit{RDFS} \rightarrow \textit{RDFS}\textsuperscript{\textit{OWL}} are similar.

\textit{DDL}\textsuperscript{\textit{OWL}} \rightarrow \mathcal{E}\textsuperscript{\textit{Co}}\textsuperscript{\textit{OWL}} is a subinstitution, because all DLL bridge rules are \mathcal{E}\textsuperscript{\textit{Co}}\textsuperscript{\textit{OWL}} bridge rules.

\textit{OWL} \rightarrow \textit{DDL}\textsuperscript{\textit{OWL}} and \textit{FOL}\textsuperscript{=} \rightarrow \mathcal{E}\textsuperscript{\textit{Co}}\textsuperscript{\textit{FOL}}= are obvious substitutions: everything is mapped into one component.

\mathcal{E}\textsuperscript{\textit{Co}}\textsuperscript{\textit{FOL}}= \rightarrow \textit{FOL}\textsuperscript{\textit{ms}}= maps each component to a sort, and function and predicate symbols are typed with the sort of their respective component.

\textbf{Simple theoroidal substitutions:} \textit{RDF} \rightarrow \textit{FOL}\textsuperscript{=}: this is a straightforward extension of \textit{bRDF} \rightarrow \textit{RDF}, axiomatising explicitly the extra conditions imposed on models. \textit{RDFS} \rightarrow \textit{FOL}\textsuperscript{=} and \textit{RDFS}\textsuperscript{\textit{OWL}} \rightarrow \textit{FOL}\textsuperscript{=} are similar. The theory of the fixed part is (after translation to \textit{FOL}\textsuperscript{=}) added to the translations of signatures.

\textit{FOL}\textsuperscript{\textit{ms}}= \rightarrow \textit{FOL}\textsuperscript{=} is a theoroidal substitution comorphism: a many-sorted signature is translated to an unsorted one by turning each sort into a unary predicate (these are called sort predicates), and each function and predicate symbol is translated by erasing its typing information in the signature, while turning it into a sentence, using the sort predicates. A sentence is translated by erasing the type information and relativising quantifiers to the sort predicates. A model is translated by turning the interpretations of sort predicates into carrier sets, and keeping functions and predicates.

\mathcal{E}\textsuperscript{\textit{Co}}\textsuperscript{\textit{OWL}} \rightarrow \textit{OWL} uses a similar technique: the different components are mapped into classes, which are then used to relativise (using intersection with these classes) sentences.

\textit{F-logic} \rightarrow \textit{FOL}\textsuperscript{=}: the additional ingredients of \textit{F-logic} are two binary relations and a bunch of partial functions; all these can be coded as (suitably axiomatised) predicates in a straightforward way. Note that the translated signatures become infinite due to the parameterisation of \textit{I}_\rightarrow \textit{etc.} over the natural numbers.
CL \rightarrow \text{CASL}: \text{ specifies the theory of lists and the implicit components of CL models explicitly in CASL.}

\text{CASL} \rightarrow \text{HOL}: \text{ codes out partiality and subsorting using standard methods, while induction axioms are translated to their explicit second-order Peano-style formulation, see [28] for details.}

\text{Rel-S} \rightarrow \text{FOL}_{\text{ms}}: \text{ database tables are mapped to predicates, and the involved datatypes are specified in FOL}_{\text{ms}}^{16}. \text{ Integrity constraints are expressible as first-order sentences, and given a first-order model, its predicates are construed as database tables.}

\textbf{Simultaneously exact and model-expansive comorphisms:} \text{ PL} \rightarrow \text{FOL}^= \text{ translates propositional variables to nullary predicates. The model translation forgets the universe (and is hence not an isomorphism). A theoroidal variant adds (to the signature translation) the axiom } \forall x, y. x = y \text{ enforcing a singleton universe (then, the model translation is at least an equivalence of categories). The translation PL} \rightarrow \text{CL} \text{ is similar. }

\text{PL} \rightarrow \text{OWL} \text{ is a each propositional variable in a signature is mapped to an atomic OWL class. Additionally, the signature translation globally adds one individual } a \text{ and the axiom } \top \subseteq \{ a \} \text{ expressing that the domain consists of a single point. A propositional sentence (i.e. a Boolean combination of propositional variables) is mapped to membership of } a \text{ in the corresponding OWL class term (i.e. a Boolean combination of atomic classes) — note that this can be expressed either as ABox statement } a : C \text{ or as TBox statement } \{ a \} \subseteq C. \text{ In order to translate an OWL model, for each atomic class } A \text{ (resulting from a propositional variable } A \text{), } a : A \text{ is evaluated, and the result is assigned to the propositional variable } A. \text{ The satisfaction condition is straightforward.}

\text{FOL}^= \rightarrow \text{CL}: \text{ the signature translation maps constants, function symbols and predicates to names. Sentences are left untouched. From a CL-model, it is possible to extract a FOL}^=\text{-model by restricting functions and predicates to those sequences that have the length of the arity of the symbol (note that this restriction is the reason for not getting an isomorphism).}

\text{bRDF} \rightarrow \text{OWL}: \text{ a bRDF signature is translated to OWL by providing a class } P \text{ and three roles } \text{sub}, \text{pred} \text{ and } \text{obj} \text{ (these reify the extension relation), and one individual per bRDF resource. A bRDF triple } (s, p, o) \text{ is translated to the OWL sentence}

\top \subseteq \exists U. (\exists sub. \{ s \} \cap \exists pred. \{ p \} \cap \exists obj. \{ o \}).

\text{From an OWL model } I, \text{ obtain a bRDF model by inheriting the universe and the interpretation of individuals (then turned into resources). The interpretation } P^I \text{ of } P \text{ gives } P_m, \text{ and } \text{EXT}_m \text{ is obtained by de-reifying, i.e.}

\text{EXT}_m(x) := \{ (y, z) | \exists u. (u, x) \in \text{pred}^I, (u, y) \in \text{sub}^I, (u, z) \in \text{obj}^I \}.

\text{RDF} \rightarrow \text{OWL} \text{ is defined similarly. The theory of RDF built-ins is (after translation to OWL) added to any signature translation. This ensures that the model translation can add the built-ins.}

\text{16} \text{Strictly speaking, for a complete specification of inductive datatypes, second-order logic is needed; in this case, the translation ends in HOL.}
OWL $\rightarrow$ RDF$^\text{OWL}$: this is the RDF serialisation of OWL, formalised as a co-
morphism in [25].

**Model-expansive comorphisms:** OWL $\rightarrow$ F-logic: translations from OWL to F-logic are discussed in [12].

5. Conclusion

We argued that there is a multitude of logics and languages in practical use for the spec-
ification of ontologies that calls for logical pluralism, understood pragmatically. In order
 to achieve ontology interoperability despite of this pluralism, it is crucial to establish and
formalise translations among these logics. We have done this, using so-called institution
comorphisms. As Proposition 16 shows, problems of logical consequence and satisfia-
bility can be translated along such translations in a sound and complete way, opening the
door for re-use of tools like theorem provers and model finders. It turns out that this is the
case even if logical consequence and satisfiability of modular ontologies is concerned:
by Proposition 16, nearly all of our translations (with the exception of some translations
of OWL to F-logic) interact well with modularity.

While we have clarified and summarised the relations among different ontology lan-
guages at the semantic level, we have not touched the methodological level. Methodol-
ogy concerns the way certain features are formalised using logic, as well as the prag-
matic level of logic. In order to make ontologies interoperable across different logics,
their methodologies (which also may vary within one logic) have to be considered as
well. Moreover, some methodologies may also lead to further logic translations that need
to be considered. This is left for future work, together with the study of more languages,
such as UML as well as some non-classical formalisms that are being used for ontologies.
Also, different modularity concepts should be studied and compared.

Acknowledgement

Work on this paper has been supported by the DFG-funded Research Center on Spatial
Cognition (SFB/TR 8).

References

Mossakowski and Kutz / The Onto-Logical Translation Graph


