The Onto-Logical Translation Graph

Till MOSSAKOWSKI [a,1] and Oliver KUTZ [b]
[a] *DFKI GmbH Bremen and University of Bremen*
[b] *Research Center on Spatial Cognition, University of Bremen*

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 applied in many different areas, including eBusiness, eHealth, eGovernment, eInclusion, eLearning, smart environments, ambient assisted living (AAL), and virtually all other information-rich applications. While the OWL standard has lead to an important unification of notation and semantics, still many diverse 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, clearly go beyond the expressiveness of OWL. 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 Common Logic, 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

[1]Corresponding Author: Till Mossakowski, DFKI GmbH Bremen, Enrique-Schmidt Strasse 8, 28359 Bremen, Germany; E-mail: Till.Mossakowski@dfki.de.

interoperable with other formalisms. 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 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.

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 [25] for an introduction.

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

- a category $\mathbf{Sign}$ of *signatures* and *signature morphisms*,
- a functor $\mathbf{Sen}\colon \mathbf{Sign} \longrightarrow \mathbf{Set}$[2] giving, for each signature Σ, the set of *sentences* $\mathbf{Sen}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *sentence translation map* $\mathbf{Sen}(\sigma)\colon \mathbf{Sen}(\Sigma) \longrightarrow \mathbf{Sen}(\Sigma')$, where often $\mathbf{Sen}(\sigma)(\varphi)$ is written as $\sigma(\varphi)$,
- a functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \longrightarrow \mathcal{CAT}$[3] giving, for each signature Σ, the category of *models* $\mathbf{Mod}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *reduct functor* $\mathbf{Mod}(\sigma)\colon \mathbf{Mod}(\Sigma') \longrightarrow \mathbf{Mod}(\Sigma)$, where often $\mathbf{Mod}(\sigma)(M')$ is written as $M'{\restriction}_\sigma$, and $M'{\restriction}_\sigma$ is called the *σ-reduct* of M', while M' is called a *σ-expansion* of $M'{\restriction}_\sigma$,
- a satisfaction relation $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ for each $\Sigma \in |\mathbf{Sign}|$,

such that for each $\sigma\colon \Sigma \longrightarrow \Sigma'$ in $\mathbf{Sign}$ the following *satisfaction condition* holds:

$$(\star) \qquad M' \models_{\Sigma'} \sigma(\varphi) \text{ iff } M'{\restriction}_\sigma \models_\Sigma \varphi$$

for each $M' \in |\mathbf{Mod}(\Sigma')|$ and $\varphi \in \mathbf{Sen}(\Sigma)$, expressing that truth is invariant under change of notation and context.[4] $\dashv$

A *theory* in an institution is a pair $\mathbf{Th} = \langle \Sigma, \Gamma \rangle$ consisting of a signature Σ and a set Γ of sentences over Σ. The models of $\mathbf{Th}$ are those Σ-models that satisfy Γ. Satisfiability and logical consequence are defined in the standard way. Moreover, the following kernel language of modular specifications [31] can be interpreted in any institution:

$$SP ::= \langle \Sigma, \Gamma \rangle \mid SP_1 \cup SP_2 \mid \sigma(SP) \mid \sigma^{-1}(SP)$$

[2] $\mathbf{Set}$ is the category having all small sets as objects and functions as arrows.

[3] $\mathcal{CAT}$ is the category of categories and functors. Strictly speaking, $\mathcal{CAT}$ is not a category but only a so-called quasicategory, which is a category that lives in a higher set-theoretic universe.

[4] Note, however, that non-monotonic formalisms can only indirectly be covered this way, but compare, e.g., [18].

with the following semantics:

$$\begin{aligned}
&\mathbf{Mod}(\langle\Sigma,\Gamma\rangle) = \{M \in \mathbf{Mod}(\Sigma) \mid M \models \Gamma\}\\
&\mathbf{Mod}(SP_1 \cup SP_2) = \mathbf{Mod}(SP_1) \cap \mathbf{Mod}(SP_2)\\
&\mathbf{Mod}(\sigma(SP)) = \{M \mid M|_\sigma \in \mathbf{Mod}(SP)\}\\
&\mathbf{Mod}(\sigma^{-1}(SP)) = \{M|_\sigma \mid M \in \mathbf{Mod}(SP)\}
\end{aligned}$$

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 [23], 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 $\mathbf{Prop}$ of propositional logic has sets Σ (propositional symbols) as signatures, and functions $\sigma : \Sigma_1 \to \Sigma_2$ between such sets as signature morphisms. A Σ-model M is a mapping from Σ to $\{true, false\}$. The reduct of a Σ_2-model M_2 along $\sigma : \Sigma_1 \to \Sigma_2$ is the Σ_1-model given by the composition $M_2 \circ \sigma$. Σ-sentences are built from Σ 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. $\dashv$

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 $\mathbf{FOL}^=$ 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 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. $\dashv$

Definition 4 (Many-sorted First-order Logic)**.** The institution $\mathbf{FOL}^{\mathsf{ms}=}$ of many-sorted first-order logic with equality is similar to 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]. $\dashv$

Although not strictly more expressive than single-sorted $\mathbf{FOL}^=$, 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)**.** Common logic has first been formalised as an institution in [23]. A common logic signature Σ (called vocabulary in Common Logic 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.[5] A Σ-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:

- 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);
- fun from UR to total functions from UD^* into UD;
- int from names in Σ to UR, such that $int(v)$ is in UD if and only if v is a discourse name;
- seq from sequence markers in Σ to UD^*.

Model reducts leave UR, UD, rel and fun untouched, while int and seq are composed with the appropriate signature morphism component. A Σ-sentence is a first-order sentence, where predications and function applications are written in a higher-order like syntax:

$$t(s)$$

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 $holds(t, s)$, and a function application $t(s)$ like the first-order term $app(t, s)$, where $holds$ and app are fictitious symbols denoting the semantic objects rel and fun. In this way, Common Logic 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 rel resp. fun in order to obtain the predicate resp. function, as discussion 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 ϕ, ψ range over a finite set Π of explicitly introduced universals. In Common Logic, this is written, using standard logical syntax (note that Common Logic is agnostic about concrete syntax)

$$\forall\phi.\Pi(\phi) \longrightarrow \phi(x)$$

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

```
(forall (?phi) (if (pi ?phi) (?phi ?x)))
```

[5]That is, a name is a discourse name if and only if its image under the signature morphism is.

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 "...")):

$$mutually\text{-}disjoint(P)$$
$$mutually\text{-}disjoint(P\ Q\ \ldots) \longleftrightarrow$$
$$(\forall x.\neg(P(x) \wedge Q(x))) \wedge mutually\text{-}disjoint(P\ \ldots) \wedge mutually\text{-}disjoint(Q\ \ldots)$$

⊣

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

Definition 6 (CASL)**.** CASL (the Common Algebraic Specification Language, [4, 29]) 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)**.** 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(**key** *id* : **integer**, *title* : **string**, *published_in* : **integer**)
journal(**key** *id* : **integer**, *name* : **string**, *impact_factor* : **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

paper[*published_in*] $\rightarrow$ *journal*[*id*] **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 $r[f] \rightarrow s[g]$ **t** is satisfied in case of **t** $=$ **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** $=$ **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: $\mathcal{OWL}$ 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

$$C ::= \mathcal{A} \,|\, \top \,|\, \bot \,|\, C_1 \sqcup C_2 \,|\, C_1 \sqcap C_2 \,|\, \neg C \,|\, \forall R.C \,|\, \exists R.C$$

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 $\mathbf{FOL}^=$. Satisfaction is the standard satisfaction of description logics.

The logic $\mathcal{SROIQ}$ [19], which is the logical core of the Web Ontology Language $\mathcal{OWL}$ 2 DL[6] extends $\mathcal{ALC}$ with the following constructs: (i) complex role boxes (denoted by $\mathcal{SR}$): these can contain: complex role inclusions such as $R \circ S \sqsubseteq S$ as well as simple role hierarchies such as $R \sqsubseteq S$, assertions for symmetric, transitive, reflexive, asymmetric and disjoint roles (called RBox sentences), as well as the construct $\exists R.\mathsf{Self}$ (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 $\mathcal{SROIQ}$ (e.g. regular role inclusions, simple roles) compare [19], and see the example given below. $\mathcal{SROIQ}$ can be straightforwardly rendered as an institutions following the previous examples, but compare also [24].

The $\mathcal{OWL}$ 2 specification contains three further DL fragments of $\mathcal{SROIQ}$, called **profiles**, namely EL, QL, and RL.[7] 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:

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

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

Apart from some exceptions[8], 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:

[6]See also `http://www.w3.org/TR/owl2-overview/`

[7]See `http://www.w3.org/TR/owl2-profiles/` for details of the specifications.

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

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[9]) borrowed from $\mathcal{OWL}$, 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.[10] This is an instance of borrowing model theory in the sense of [9], by which an institution OBO1.4-OWL is obtained. Since the translation is still partial, OBO1.4-OWL is only a subset of the full OBO 1.4. For instance, in order to preserve decidability, $\mathcal{SROIQ}$ 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 $\mathcal{SROIQ}$. 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. $\dashv$

Definition 10 (RDF and RDFS)**.** Following [24], 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* $\mathbf{R_s}$ in bRDF is a set of *resource references*. For $sub, pred, obj \in \mathbf{R_s}$, 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 $\mathbf{R_s}$-model $M = \langle R_m, P_m, S_m, EXT_m \rangle$ consists of a *set* R_m *of resources*, a set $P_m \subseteq R_m$ of predicates, a *mapping function* $S_m : \mathbf{R_s} \to R_m$, and an *extension function* $EXT_m : P_m \to \mathcal{P}(R_m \times R_m)$ mapping every predicate to a set of pairs of resources. Satisfaction is defined as follows:

$$\mathfrak{M} \models_{\mathbf{R_s}} (sub, pred, obj) \Leftrightarrow (S_m(sub), (S_m(obj)) \in EXT_m(S_m(pred)).$$

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 `rdf:type` and `rdf:Property` and `rdf:subject`, and sentences (`rdf:type`, `rdf:type`, `rdf:Property`), and (`rdf:subject`, `rdf:type`, `rdf:Property`).

In the models, the standard vocabulary is interpreted with a fixed model. Moreover, for each RDF-model $M = \langle R_m, P_m, S_m, EXT_m \rangle$, if $p \in P_m$, then it must hold

[9]Some language constructs, such as 'being necessarily false' were seen to not have sufficiently clear semantics, and were subsequently dropped from the OBO language.

[10]`http://www.geneontology.org/GO.format.obo-1_4.shtml#OWL`

$(p, S_m(\texttt{rdf:Property})) \in EXT_m(\texttt{rdf:type})$. 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 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-OWL, an extension of RDFS with resources like owl:Thing and owl:oneOf, tailored towards the representation of $\mathcal{OWL}$.

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) [6] 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 $\mathcal{ALC}$, the basic link language of $\mathcal{E}$-connections is $\mathcal{ALCI}^{\mathsf{ms}}$.[11]

The idea to 'connect' logics can be elegantly generalised to the institutional level (compare [3] who note that their 'connections' are an instance of a more general co-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 $\mathcal{S}_1$ and $\mathcal{S}_2$ 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 [22], the DDLs of [6], as well as the connections of Baader and Ghilardi [3] 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 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-OWL is the institution whose component logics are $\mathcal{OWL}$ based and whose bridge rules follow DDL restrictions, the institution of EConn-OWL allows only $\mathcal{OWL}$-based components, but allows the more general bridge expressivity of $\mathcal{E}$-connections,[12] and EConn-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. $\mathcal{E}$-connections of $\mathcal{OWL}$ ontologies can be encoded within 'many-sorted' $\mathcal{SROIQ}$, with additional syntactic restrictions capturing the allowed bridge axioms, see also Sec. 4. $\dashv$

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

[11] But can of course be weakened to $\mathcal{ALC}^{\mathsf{ms}}$ or sub-Boolean DL, or indeed strengthened to more expressive many-sorted DLs involving e.g. number restrictions or Boolean operators on links, see [22] for details.

[12] Note that allowing full $\mathcal{OWL}$ expressivity on the link language leads to undecidability also for $\mathcal{OWL}$-based components.

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 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[13].

Method expression have the following forms:

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

Here, *ScalarMethod*, *ScalarMethod* 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 ε, and :: with an irreflexive partial order $\prec$, such that $a \varepsilon B$ and $b \preceq c$ imply $a \varepsilon c$,
- for $u \in U$ and each $n \geq 0$, there are
  - ∗ partial functions $I^n_{\rightarrow}(u), I^n_{\bullet\rightarrow}(u) \colon U^{n+1} \mathrel{\circ\!\!\rightarrow} U$,
  - ∗ partial functions $I^n_{\twoheadrightarrow}(u), I^n_{\bullet\twoheadrightarrow}(u) \colon U^{n+1} \mathrel{\circ\!\!\rightarrow} \mathcal{P}(U)$,
  - ∗ partial anti-monotonic functions $I^n_{\Rightarrow}(u), I^n_{\Rrightarrow}(u) \colon U^{n+1} \mathrel{\circ\!\!\twoheadrightarrow} \mathcal{P}_{\uparrow}(U)$, where $\mathcal{P}_{\uparrow}(U)$ 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 ε and $\prec$, respectively. An object atom $O[\mathit{ScalarMethod} \,@\, t_1, \ldots t_n \rightarrow t]$ holds in a model M under a variable valuation ν, if $I^n_{\rightarrow}(\nu(\mathit{ScalarMethod}))(\nu(O), \nu(t_1), \ldots, \nu(t_n))$ is defined and equal to $\nu(t)$; similarly for $\bullet\!\!\rightarrow$. $O[\mathit{SetMathod} \,@\, t_1, \ldots t_n \twoheadrightarrow \{u_1, \ldots, u_m\}]$ holds in M w.r.t. ν, if $I^n_{\twoheadrightarrow}(\nu(\mathit{SetMethod}))(\nu(O), \nu(t_1), \ldots, \nu(t_n))$ is defined and contains the set $\{\nu(u_1), \ldots, \nu(u_m)\}$; similarly for $\bullet\!\!\twoheadrightarrow$, $\Rightarrow$ and $\Rrightarrow$.

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, the rationale and the methodology of use of F-logic in the field of object-oriented modelling, see [20]. $\dashv$

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 [30] (one additional feature of Isabelle are type classes).

[13] object molecules $O[me_1; \ldots; me_n]$ abbreviate conjunctions of object atoms.

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, 28]:

Definition 14 (Institution Comorphism)**.** Given two institutions I and J with $I = (\mathbf{Sign}, \mathbf{Mod}, \mathbf{Sen}, \models)$ and $J = \big(\mathbf{Sign}', \mathbf{Mod}', \mathbf{Sen}', \models'\big)$, an **institution comorphism** from I to J consists of a functor $\Phi : \mathbf{Sign} \longrightarrow \mathbf{Sign}'$, and natural transformations $\beta : \mathbf{Mod}' \circ \Phi \Longrightarrow \mathbf{Mod}$ and $\alpha : \mathbf{Sen} \Longrightarrow \mathbf{Sen}' \circ \Phi$, such that the *satisfaction condition*

$$M' \models^{I'}_{\Phi(\Sigma)} \alpha_\Sigma(\varphi) \Leftrightarrow \beta_\Sigma(M') \models^{I}_{\Sigma} \varphi.$$

holds.

Here, $\Phi(\Sigma)$ is the translation of signature Σ from institution I to institution J, $\alpha_\Sigma(\varphi)$ is the translation of the Σ-sentence φ to a $\Phi(\Sigma)$-sentence, and $\beta_\Sigma(M')$ is the translation (or perhaps better: reduction) of the $\Phi(\Sigma)$-model M' to a Σ-model.

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

A simple example is given by considering the well-known translation of $\mathcal{OWL}$ 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 after introducing the details of the institution-based formalisation in the next section.

The practical usefulness of institution comorphisms grows with their properties:

Definition 15. An institution comorphism is *model-expansive*, if each model translation β_Σ is surjective on objects.

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

$$\begin{array}{ccc}
\mathbf{Mod}^I(\Sigma_2) & \xleftarrow{\beta_{\Sigma_2}} & \mathbf{Mod}^J(\Phi(\Sigma_2)) \\
\Big\downarrow {\scriptstyle \mathbf{Mod}^I(\sigma)} & & \Big\downarrow {\scriptstyle \mathbf{Mod}^J(\Phi(\sigma))} \\
\mathbf{Mod}^I(\Sigma_1) & \xleftarrow{\beta_{\Sigma_1}} & \mathbf{Mod}^J(\Phi(\Sigma_1))
\end{array}$$

admits (weak) amalgamation, i.e. any for any two models $M_2 \in \mathbf{Mod}^I(\Sigma_2)$ and $M_1' \in \mathbf{Mod}^J(\Phi(\Sigma_1))$ with $M_2|_\sigma = \beta_{\Sigma_1}(M_1')$, there is a unique (not necessarily unique) $M_2' \in \mathbf{Mod}^J(\Phi(\Sigma_2))$ with $\beta_{\Sigma_2}(M_2') = M_2$ and $M_2'|_{\Phi(\sigma)} = M_1'$. In case that $\mathcal{D}$ consists of all signature morphisms, the (weak) $\mathcal{D}$-amalgamation property is also called (weak) *exactness*.

An institution (co)morphism $\mu = (\Phi, \alpha, \beta) \colon I \longrightarrow J$ is said to be *model-isomorphic* if for each $\Sigma \in \mathbf{Sign}^I$, β_Σ is an isomorphism. It is a *subinstitution comorphism* [26], if moreover Φ is an embedding and each α_Σ is injective. The intuition is that theories should be embedded, while models should be represented exactly (such that model-theoretic results carry over).

It is easy to see that a model-isomorphic comorphisms also is model-expansive and exact.

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

$$\Gamma \models^I_\Sigma \varphi \Longrightarrow \alpha_\Sigma(\Gamma) \models^J_{\Phi(\Sigma)} \alpha_\Sigma(\varphi),$$

$$\Gamma \textit{ satisfiable } \Longleftarrow \alpha_\Sigma(\Gamma) \textit{ satisfiable},$$

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

$$SP \models^I_\Sigma \varphi \Longleftrightarrow \mu(SP) \models^J_{\Phi(\Sigma)} \alpha_\Sigma(\varphi),$$

$$SP \textit{ satisfiable } \Longleftrightarrow \mu(SP) \textit{ satisfiable},$$

where $\mu(SP)$ is the translation of SP using Φ and α.

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 $\mathcal{OWL}$ ontologies. They use so-called bridging axioms (formulated in first-order) to relate the meaning of terms in different ontologies,[14] 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, which essentially 'coincides' with the direct semantics of $\mathcal{OWL}$, 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:

Subinstitutions: $PL \to FOL$, $EL \to OWL$, $QL \to OWL$ and $RL \to OWL$ and $FOL \to FOL\text{-}ms$ are obvious subinstitutions.

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

$E\text{-}Conn\text{-}OWL \to E\text{-}Conn\text{-}FOL$ uses $OWL \to FOL$ twice, at the level of the base logic and at the level of the bridge rules.

[14] Not to be confused with the 'bridge axioms' in DDL [6].

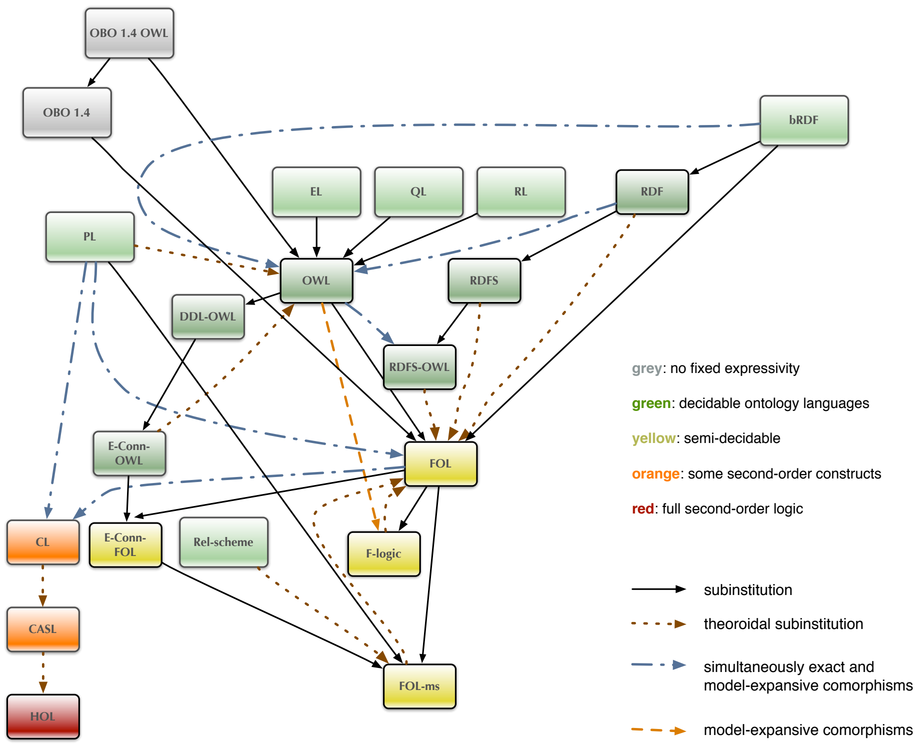

Figure 1. Logic translations between ontology languages

$PL \to FOL\text{-}ms$ is a subinstitution by mapping propositional variables to nullary predicates.

$OBO1.4\text{-}OWL \to OWL$: signatures and sentences are translated according to the OBO standard, whereas the model translation is the identity (due to borrowing of model theory).

$OBO1.4 \to FOL$ extends the composition $OBO1.4\text{-}OWL \to OWL \to FOL$ by an explicit straight-forward coding of the additional features not present in $OBO1.4\text{-}OWL$.

$bRDF \to FOL$ The subinstitution comorphism from bRDF to FOL maps a a bRDF signature $\mathbf{R_s}$ to the FOL signature $\Phi(\mathbf{R_s})$ which has $\mathbf{R_s}$ as set of constants, and moreover is equipped with a unary predicate P and a ternary predicate EXT. A bRDF-sentence $(sub, pred, obj)$ is translated to $EXT(sub, pred, obj)$. Finally, a FOL-model of $\Phi(\mathbf{R_s})$ is translated to the bRDF which has the model's universe as set of resources R_m, while P_m is given by the interpretation of P and S_m by the interpretation of the constants. EXT_m can be easily constructed from the interpretation of EXT.

$FOL \to F\text{-}logic$ is an obvious subinstitution.

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

DDL-OWL $\rightarrow$ E-$Conn$-OWL is a subinstitution, because all DLL bridge rules are E-$Conn$ bridge rules.

$OWL \rightarrow DDL$-OWL and $FOL \rightarrow E$-$Conn$-FOL are obvious subinstitutions: everything is mapped into one component.

E-$Conn$-FOL $\rightarrow$ FOL-ms maps each component to a sort, and function and predicates symbols are typed with the sort of their respective component.

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

$RDF \rightarrow FOL$: this is a straightforward extension of $bRDF \rightarrow RDF$, axiomatising explicitly the extra conditions imposed on models. $RDFS \rightarrow FOL$ and $RDFS$-$OWL \rightarrow FOL$ are similar. The theory of the fixed part is (after translation to FOL) added to the translations of signatures.

FOL-ms $\rightarrow$ FOL is a theoroidal subinstitution 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 turing the interpretations of sort predicates into carrier sets, and keeping functions and predicates.

E-$Conn$-OWL $\rightarrow$ OWL uses a similar technique: the different components are mapped into classes, which are then used to relativise (using intersection with these classes) sentences.

F-logic$\rightarrow$ FOL: the additional ingredients of 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 $I_{\rightarrow}$ etc. over the natural numbers.

CL $\rightarrow$ CASL: specifies the theory of lists and the implicit components of CL models explicitly in CASL.

CASL $\rightarrow HOL$ codes out partiality and subsorting using standard methods, while induction axioms are translated to their explicit second-order Peano-style formulation, see [27] for details.

Rel-$scheme \rightarrow FOL$-ms: database tables are mapped to predicates, and the involved datatypes are specified in FOL[15]. Integrity constraints are expressible as first-order sentences, and given a first-order model, its predicates are construed as database tables.

[15] Strictly speaking, for a complete specification of inductive datatypes, second-order logic is needed; in this case, the translation ends in HOL.

Simultaneously exact and model-expansive comorphisms: $PL \to FOL$ 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 $\forall x, y \,.\, x = y$ enforcing a singleton universe (then, the model translation is at least an equivalence of categories). The translation $PL \to CL$ is similar.

$FOL \to CL$: 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-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).

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

$$\top \sqsubseteq \exists U.(\exists sub.\{s\} \sqcap \exists pred.\{p\} \sqcap \exists obj.\{o\}).$$

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

$$EXT_m(x) := \{(y, z) | \exists u.(u, x) \in pred^{\mathcal{I}}, (u, y) \in sub^{\mathcal{I}}, (u, z,) \in obj^{\mathcal{I}}\}.$$

$RDF \to OWL$ 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.

$OWL \to RDFS\text{-}OWL$: this is the RDF serialisation of OWL, formalised as a comorphism in [24].

Model-expansive comorphisms: $OWL \to$ *F-logic*: translations from $\mathcal{OWL}$ 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 specification 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 satisfiability 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 $\mathcal{OWL}$ to F-logic) interact well with modularity.

While we have clarified and summarised the relations among different ontology languages at the semantic level, we have not touched the methodological level. Methodology concerns the way certain features are formalised using logic, as well as the pragmatic 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.

Acknowledgements

Work on this paper has been supported by the DFG-funded collaborative research centre SFB/TR 8 'Spatial Cognition' and by the German Federal Ministry of Education and Research (Project 01 IW 07002 FormalSafe).

References

[1] ADÁMEK, J., HERRLICH, H., AND STRECKER, G. *Abstract and Concrete Categories*. Wiley, New York, 1990. Freely available at `http://www.math.uni-bremen.de/~dmb/acc.pdf`.

[2] BAADER, F., CALVANESE, D., MCGUINNESS, D., NARDI, D., AND PATEL-SCHNEIDER, P. F., Eds. *The Description Logic Handbook*. Cambridge University Press, 2003.

[3] BAADER, F., AND GHILARDI, S. Connecting Many-Sorted Theories. *The Journal of Symbolic Logic 72*, 2 (2007), 535–583.

[4] BIDOIT, M., AND MOSSES, P. D. CASL *User Manual*, vol. 2900 of *Lecture Notes in Computer Science*. Springer, 2004. Freely available at `http://www.cofi.info`.

[5] BORGIDA, A. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence 82*, 1–2 (1996), 353–367.

[6] BORGIDA, A., AND SERAFINI, L. Distributed Description Logics: Assimilating Information from Peer Sources. *Journal of Data Semantics 1* (2003), 153–184.

[7] BORZYSZKOWSKI, T. Higher-order logic and theorem proving for structured specifications. In *WADT* (1999), D. Bert, C. Choppy, and P. D. Mosses, Eds., vol. 1827 of *Lecture Notes in Computer Science*, Springer, pp. 401–418.

[8] BORZYSZKOWSKI, T. Logical systems for structured specifications. *Theoretical Computer Science 286* (2002), 197–245.

[9] CERIOLI, M., AND MESEGUER, J. May I borrow your logic? (transporting logical structures along maps). *Theoretical Computer Science 173* (1997), 311–347.

[10] CHURCH, A. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic 5*, 1 (1940), 56–69.

[11] COMMON LOGIC WORKING GROUP. Common Logic: Abstract syntax and semantics. Tech. rep., 2003.

[12] DE BRUIJN, J., AND HEYMANS, S. On the Relationship between Description Logic-based and F-Logic-based Ontologies. *Fundam. Inf. 82* (August 2008), 213–236.

[13] DIACONESCU, R. *Institution-independent Model Theory*. Birkhäuser, 2008.

[14] DOU, D., AND MCDERMOT, D. Towards theory translation. In *Declarative Agent Languages and Technologies IV* (2007), Springer.

[15] GOGUEN, J. A., AND BURSTALL, R. M. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the ACM 39* (1992), 95–146.

[16] GOGUEN, J. A., AND ROŞU, G. Institution morphisms. *Formal aspects of computing 13* (2002), 274–307.

[17] GOLBREICH, C., HORRIDGE, M., HORROCKS, I., MOTIK, B., AND SHEARER, R. OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences. In *Proc. of the 6th Int. Semantic Web Conference (ISWC 2007)* (Busan, Korea, November 11-15 2007), K. A. et al., Ed., vol. 4825 of *LNCS*, Springer, pp. 169–182.

[18] GUERRA, S. Composition of Default Specifications. *J. Log. Comput. 11*, 4 (2001), 559–578.

[19] HORROCKS, I., KUTZ, O., AND SATTLER, U. The Even More Irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)* (June 2006), AAAI Press, pp. 57–67.

[20] KIFER, M., LAUSEN, G., AND WU, J. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM 42* (July 1995), 741–843.

[21] KUTZ, O., LÜCKE, D., AND MOSSAKOWSKI, T. Heterogeneously Structured Ontologies—Integration, Connection, and Refinement. In *Advances in Ontologies. Proc. of the KR-08 Ontology Workshop (KROW 2008)* (Sydney, Australia, 2008), T. Meyer and M. A. Orgun, Eds., vol. 90 of *CRPIT*, ACS, pp. 41–50.

[22] KUTZ, O., LUTZ, C., WOLTER, F., AND ZAKHARYASCHEV, M. $\mathcal{E}$-Connections of Abstract Description Systems. *Artificial Intelligence 156*, 1 (2004), 1–73.

[23] KUTZ, O., MOSSAKOWSKI, T., AND LÜCKE, D. Carnap, Goguen, and the Hyperontologies: Logical Pluralism and Heterogeneous Structuring in Ontology Design. *Logica Universalis 4*, 2 (2010), 255–333. Special Issue on 'Is Logic Universal?'.

[24] LUCANU, D., LI, Y.-F., AND DONG, J. S. Semantic Web Languages—Towards an Institutional Perspective. In *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday* (2006), K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, Eds., vol. 4060 of *Lecture Notes in Computer Science*, Springer, pp. 99–123.

[25] MAC LANE, S. *Categories for the Working Mathematician*, 2nd ed. Springer, Berlin, 1998.

[26] MESEGUER, J. General logics. In *Logic Colloquium 87*. North Holland, 1989, pp. 275–329.

[27] MOSSAKOWSKI, T. Relating casl with other specification languages: the institution level. *Theoretical Computer Science 286* (2002), 367–475.

[28] MOSSAKOWSKI, T., TARLECKI, A., AND DIACONESCU, R. What is a logic translation? *Logica Universalis 3*, 1 (2009), 95–124. Winner of the Universal Logic 2007 Contest.

[29] MOSSES, P. D., Ed. CASL *Reference Manual*, vol. 2960 of *Lecture Notes in Computer Science*. Springer, 2004. Freely available at `http://www.cofi.info`.

[30] NIPKOW, T., PAULSON, L. C., AND WENZEL, M. *Isabelle/HOL—A Proof Assistant for Higher-Order Logic*, vol. 2283 of *LNCS*. Springer, 2002.

[31] SANNELLA, D., AND TARLECKI, A. Specifications in an arbitrary institution. *Information and Computation 76* (1988), 165–210.