

Heterogeneously Structured Ontologies

Integration, Connection, and Refinement

Oliver Kutz¹

Dominik Lücke¹

Till Mossakowski^{1,2}

¹ SFB/TR 8 Spatial Cognition, University of Bremen
Cartesium, Enrique-Schmidt Strasse
28359, Bremen, Germany
Email: {okutz,luecke}@informatik.uni-bremen.de

² DFKI GmbH, Bremen, Germany
Cartesium, Enrique-Schmidt Strasse
28359, Bremen, Germany
Email: Till.Mossakowski@dfki.de

Abstract

This paper systematically applies tools and techniques from the area of algebraic specification theory to corresponding ontology structuring and design tasks.

We employ the heterogeneous structuring mechanisms of the heterogeneous algebraic specification language HETCASL for defining an abstract notion of structured heterogeneous ontology. This approach enables the designer to split up a heterogeneous ontology into semantically meaningful parts and employ dedicated reasoning tools to them.

In particular, we distinguish three fundamentally different kinds of combining heterogeneous ontologies: integration, connection, and refinement.

Keywords: Ontology Design & Reasoning; Modularity; Combination Techniques; Algebraic Specification

1 Introduction

Ontologies play an increasingly important role in various areas of Knowledge Representation ranging from the life sciences and engineering domains to linguistic semantics. In the process, ontologies are being designed in a broad spectrum of logics, with considerably varying expressivity and supporting quite different reasoning methods. Logics being used include description logics like *SROIQ(D)* (Horrocks et al. 2006), relational database schemes, as well as first-order and modal logics. While modularity, aligning and re-using (parts of) ontologies has received considerable attention recently,¹ there is little work on formal structuring and the aspect of heterogeneity.

We believe that a lot can be learned in this respect from techniques developed for (algebraic) specification in software engineering, and provide a systematic account that parallels structuring techniques from algebraic specification with typical problems found in ontology design.

Our work builds on and generalises and extends ideas of (Bench-Capon & Malcolm 1999), (Alagić &

Bernstein 2002), (Madhavan et al. 2002), and in particular (Goguen 2005, 2006). In recent publications, we have discussed the problem of inheriting conservativity properties from parts (or modules) of an ontology to an overall integration obtained through a colimit operation (Kutz & Mossakowski 2007, 2008), and analysed various alignment approaches from the institutional viewpoint (Kutz et al. 2008).

This paper addresses the following:

(1) we develop a rather abstract view of heterogeneously structured ontologies encompassing essentially all logics used in ontology design today and allowing to model the most complex relationships between various ontologies; (2) we systematise the field of ‘combining ontologies’ by identifying three classes of such combinations: *integrations*, *connections*, and *refinements*. The differentiating criteria are the use of signatures in the overall combination and the corresponding model-theoretic properties; (3) we analyse how various well-known ontology design and combination techniques fit into these abstract categories, including structuring through conservative extensions, ontology alignments, \mathcal{E} -connections, and database scheme–ontology reconciliation; (4) finally, the appendix contains a discussion how the Heterogeneous Tool Set (HETS) can support various reasoning and ontology engineering tasks; we also indicate the current and planned tool support for existing ontology languages and reasoners.

2 Heterogeneous Ontologies and Structuring

The study of modularity principles can be carried out to a quite large extent independently of the details of the underlying logical system that is used. The notion of **institutions** was introduced by Goguen and Burstall in the late 1970s exactly for this purpose (see (Goguen & Burstall 1992)). They capture in a very abstract and flexible way the notion of a logical system by describing how, in any logical system, signatures, models, sentences (axioms) and satisfaction (of sentences in models) are related.

The importance of the notion of institutions lies in the fact that a surprisingly large body of logical notions and results can be developed in a way that is completely independent of the specific nature of the underlying institution.²

We assume some acquaintance with the basic notions of category theory and refer to (Adámek et al. 1990) or (Mac Lane 1998) for an introduction. If \mathcal{C} is a category, \mathcal{C}^{op} is the dual category where all arrows

Copyright ©2008, Australian Computer Society, Inc. This paper appeared at the Knowledge Representation Ontology Workshop (KROW 2008), Sydney, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 90, Thomas Meyer and Abhaya Nayak, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹For work on modularity in ontologies compare the workshop series (Haase et al. 2006, Cuenca Grau et al. 2007, Sattler & Tamin 2008), and for work on ontology alignment/matching compare the textbook (Euzenat & Shvaiko 2007), as well as the workshops of the ‘Ontology Alignment Evaluation Initiative’ (see <http://oaei.ontologymatching.org/>).

²For an extensive treatment of model theory in this setting, see (Diaconescu 2008).

are reversed. For a category \mathcal{C} , we denote by $|\mathcal{C}|$ the class of its objects.

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

- a category **Sign** of *signatures*,
- a functor $\mathbf{Sen}: \mathbf{Sign} \rightarrow \mathbf{Set}^3$ giving, for each signature Σ , the set of *sentences* $\mathbf{Sen}(\Sigma)$, and for each signature morphism $\sigma: \Sigma \rightarrow \Sigma'$, the *sentence translation map* $\mathbf{Sen}(\sigma): \mathbf{Sen}(\Sigma) \rightarrow \mathbf{Sen}(\Sigma')$, where often $\mathbf{Sen}(\sigma)(\varphi)$ is written as $\sigma(\varphi)$,
- a functor $\mathbf{Mod}: \mathbf{Sign}^{op} \rightarrow \mathcal{CAT}^4$ giving, for each signature Σ , the category of *models* $\mathbf{Mod}(\Sigma)$, and for each signature morphism $\sigma: \Sigma \rightarrow \Sigma'$, the *reduct functor* $\mathbf{Mod}(\sigma): \mathbf{Mod}(\Sigma') \rightarrow \mathbf{Mod}(\Sigma)$, where often $\mathbf{Mod}(\sigma)(M')$ is written as $M' \upharpoonright_{\sigma}$, and $M' \upharpoonright_{\sigma}$ is called the σ -*reduct* of M' , while M' is called a σ -*expansion* of $M' \upharpoonright_{\sigma}$,
- a satisfaction relation $\models_{\Sigma} \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ for each $\Sigma \in |\mathbf{Sign}|$,

such that for each $\sigma: \Sigma \rightarrow \Sigma'$ in **Sign** the following *satisfaction condition* holds:

$$(\star) \quad M' \models_{\Sigma'} \sigma(\varphi) \text{ iff } M' \upharpoonright_{\sigma} \models_{\Sigma} \varphi$$

for each $M' \in |\mathbf{Mod}(\Sigma')|$ and $\varphi \in \mathbf{Sen}(\Sigma)$, expressing that truth is invariant under change of notation and enlargement of context.⁵ \diamond

Examples of institutions include, among others, first- and higher-order classical logic, description logics, and various (quantified) modal logics:

Example 1. First-order Logic. In the institution $\mathbf{FOL}^{ms=}$ of many-sorted first-order logic with equality, signatures are many-sorted first-order signatures, consisting of sorts and typed function and predicate symbols. Signature morphisms map symbols such that typing is preserved. Models are many-sorted first-order structures. 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. \diamond

Example 2. Relational Schemes. A signature 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. Signature morphisms map sorts, relation symbols and field names. A model consists of a carrier set for each sort, and an n -ary relation for each relation symbol with n fields. A model reduction just forgets the parts of a model that are not needed. A sentence is a link (integrity constraint) between two field names of two relation symbols. Sentence translation is just renaming. A link is satisfied in a model if for each element occurring in the source field component of a tuple in the source relation, the same element also occurs in the target field component of a tuple in the target relation. \diamond

³**Set** is the category having all small sets as objects and functions as arrows.

⁴ \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.

⁵Note, however, that non-monotonic formalisms can only indirectly be covered this way, but compare, e.g., (Guerra 2001).

Example 3. Description Logics. Signatures of the description logic \mathcal{ALC} consist of a set of B of atomic concepts and a set R of roles, 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

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

Sentence translation and reduct is defined similarly as in $\mathbf{FOL}^=$. Satisfaction is the standard satisfaction of description logics. \mathcal{ALC}^{ms} is the many-sorted variant of \mathcal{ALC} . \mathcal{ALCO} is obtained from \mathcal{ALC} by extending signatures with nominals.

The (sub-Boolean) description logic \mathcal{EL} restricts \mathcal{ALC} as follows: $C ::= B \mid \top \mid C_1 \sqcap C_2 \mid \exists R.C$. \mathcal{SHOIN} extends \mathcal{ALC} with role hierarchies, transitive and inverse roles, (unqualified) number restrictions, and nominals, etc. \diamond

Example 4. Modal Logics. The modal logic $\mathbf{S4}_u$ has signatures as classical propositional logic, consisting of propositional variables. Sentences are built as in propositional logic, but add two unary modal operators, \square and \blacksquare . Models are standard Kripke structures but based on reflexive and transitive relations. Satisfaction is standard modal satisfaction, where \square is interpreted by the transitive reflexive relation, and \blacksquare by universal quantification over worlds.

The standard formulation of first-order modal logic $\mathbf{QS5}$ (due to Kripke) has signatures similar to $\mathbf{FOL}^=$, including variables and predicate symbols. Sentences follow the grammar for $\mathbf{FOL}^=$ -sentences using Booleans, quantifiers, and identity, while adding the \square operator, but leaving out constants and function symbols. Models are constant-domain first-order Kripke structures, with the usual first-order modal satisfaction. \diamond

2.1 Structured Ontologies

The essential advantage of the theory of institutions is the possibility of providing structuring operations and module concepts independently of the underlying logical system. Hence, in the sequel, let us fix some arbitrary institution $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$. The basic structuring operation for ontologies is surely that of *importing* other ontologies. The notion of *development graph* captures this, and also renaming of symbols.

Definition 2. A **development graph** is an acyclic, directed graph⁶ $\mathcal{DG} = \langle \mathcal{N}, \mathcal{L} \rangle$. Here, \mathcal{N} is a set of nodes. Each node $N \in \mathcal{N}$ is labelled with a pair (Σ^N, Ψ^N) such that Σ^N is a signature and $\Psi^N \subseteq \mathbf{Sen}(\Sigma^N)$ is the set of **local axioms** of N . \mathcal{L} is a set of directed links, so-called (global⁷) **definition**

links $K \xrightarrow{\sigma} N$, annotated with a signature morphism $\sigma: \Sigma^K \rightarrow \Sigma^N$.

Given a node $N \in \mathcal{N}$, its associated theory $\mathbf{Th}_{\mathcal{DG}}(N)$ is inductively defined to consist of

- all the local axioms Ψ^N , and
- for each $K \xrightarrow{\sigma} N \in \mathcal{DG}$, all of $\mathbf{Th}_{\mathcal{DG}}(K)$ translated by σ .

The class of models $\mathbf{Mod}_{\mathcal{DG}}(N)$ of a node N is defined as $\mathbf{Mod}(\mathbf{Th}_{\mathcal{DG}}(N))$. \diamond

⁶In (Kutz & Mossakowski 2007, 2008), we have identified a structured ontology with a diagram.

⁷There are also local and hiding definition links, which require a more refined model-theoretic semantics.

Complementary to definition links, which *define* the theories of related nodes, we also allow for **theorem links** with the help of which we are able to *postulate* relations between different theories. A

(global) theorem link is an edge $K \xrightarrow{\sigma} N$, where $\sigma: \Sigma^K \rightarrow \Sigma^N$. \mathcal{DG} implies a theorem link $K \xrightarrow{\sigma} N$ (denoted $\mathcal{DG} \models K \xrightarrow{\sigma} N$) iff for all $M \in \mathbf{Mod}_{\mathcal{DG}}(N)$, $M|_{\sigma} \in \mathbf{Mod}_{\mathcal{DG}}(K)$.

A global definition (or also theorem) link $K \xrightarrow{\sigma} N$ can be strengthened to a **conservative extension link** (denoted as $K \xrightarrow[\text{cons}]{\sigma} N$);

it holds if every K -model has a σ -expansion to an N -model. Such annotations can be seen as another kind of proof obligations. Definitional extensions are introduced in a similar way (annotated with *def*); here the σ -expansion has to be unique.

Many languages for structuring, modularity and alignment of ontologies can be mapped into this formalism of development graphs.⁸ In this paper, we use the language CASL (Bidoit & Mosses 2004). A CASL library consists of specification definitions of the form

```
spec <name> =
  <spec>
end
```

A specification `<spec>` can be a basic specification consisting of a signature and some axioms (with syntax specific to the given institution). It corresponds to a node with local axioms in a development graph. Concerning structuring, specifications can be extended or united, written `<spec> then <spec>` or `<spec> and <spec>`, and leading to definition links in the development graph. Extensions add some new signature elements and axioms, and can be declared to be conservative or definitional. Unions unite the requirements of two specifications, thereby intersecting their model classes. Renamings, written `<spec> with <signature-morphism>`, rename a specification along a signature morphism; again, this leads to a definition link in the development graph. The declaration `view view1: sp1 to sp2` will generate a theorem link between the nodes representing `sp1` and `sp2` in the development graph. Details of the translation to development graphs, as well as a treatment of hiding, can be found in (Mossakowski et al. 2006). Moreover, the appendix contains additional details about the syntax of CASL (and HETCASL introduced below) as well as example specifications.

2.2 Heterogeneous Ontologies

As (Schorlemmer & Kalfoglou 2008) argue convincingly, since ontologies are being written in many different formalisms, like relation schemata, description logics, first-order logic, and modal (first-order) logics, alignments of ontologies need to be constructed across different institutions.

To obtain heterogeneous logical theories, one needs to fix some graph of logics and logic translations, usually formalised as institutions and so-called institution comorphisms, see (Goguen & Roşu 2002). The latter are again governed by the satisfaction condition, this time expressing that truth is invariant also under change of notation across different logical formalisms:

⁸By ‘modularity’ we here refer to the notion of ‘ontological module’ defined through conservativity properties, as it has been investigated for instance in Konev et al. (2008), Cuenca Grau, Horrocks, Kazakov & Sattler (2008), Kutz & Mossakowski (2008).

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

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: reduction) of the $\Phi(\Sigma)$ -model M' to a Σ -model.

The so-called **Grothendieck institution** is a technical device for giving a semantics to heterogeneous theories involving several institution (see Diaconescu 2002, Mossakowski 2002). The Grothendieck institution is basically a flattening, or disjoint union, of the logic graph. A signature in the Grothendieck institution consists of a pair (L, Σ) where L is a logic (institution) and Σ is a signature in the logic L . Similarly, a Grothendieck signature morphism $(\rho, \sigma): (L_1, \Sigma_1) \rightarrow (L_2, \Sigma_2)$ consists of a logic translation $\rho = (\Phi, \alpha, \beta): L_1 \rightarrow L_2$ plus an L_2 -signature morphism $\sigma: \Phi(\Sigma_1) \rightarrow \Sigma_2$. Sentences, models and satisfaction in the Grothendieck institution are defined in a componentwise manner.

We now arrive at the following:

Definition 3. An **abstract structured heterogeneous ontology** (w.r.t. some logic graph) is a node O in a development graph \mathcal{DG} in the corresponding Grothendieck institution. We sometimes also identify O with its theory $\mathbf{Th}_{\mathcal{DG}}(O)$; however, note that then the structuring is lost. \diamond

To be able to write down such heterogeneous ontologies in a concise manner, we extend CASL to HETCASL as follows: HETCASL provides the notation `logic <logic-name>`, which defines the institution of the following specifications until that keyword occurs again. Also, a specification can be translated along a comorphism; this is written `<spec> with logic <comorphism-name>`. See the extended example on Page 7 for the look-and-feel of HETCASL specifications. Of course, abstract structured heterogeneous ontologies can be formulated in different notations, and HETCASL is only one of them. Another option would be an extension of OWL structuring mechanisms by keywords dealing with heterogeneity.

3 Heterogeneous Integration

Informally, an *integration* of two ontologies O_1, O_2 into a third ontology O is any operation by which O_1, O_2 are ‘re-interpreted’ from the (global) point of view of O .

Definition 4. Given ontologies O_1, O_2 , and an ontology O , in institutions I_1, I_2 and I , respectively, we say that O **heterogeneously integrates** O_1 and O_2 if there are theorem links (i.e. theory interpretations) $\lambda_1: O_1 \mapsto O$ and $\lambda_2: O_2 \mapsto O$. \diamond

Thus, given $O_1 \models_{I_1} \phi$ and $O_2 \models_{I_2} \psi$, we have $O \models_I \lambda_1(\phi), \lambda_2(\psi)$, i.e., consequence is preserved upwards.

In the approach of (Schorlemmer & Kalfoglou 2008), two ontologies O_1 and O_2 are aligned by mapping them into a *common reference ontology* O as follows: theories O_1 and O_2 are said to be **semantically integrated** with respect to a theory O if (1) there exist *consequence-preserving sentence translations* $\alpha_1: O_1 \rightarrow O, \alpha_2: O_2 \rightarrow O$; (2) there exist *structure reducts* $\beta_1: \mathbf{Mod}(O) \rightarrow \mathbf{Mod}(O_1), \beta_2: \mathbf{Mod}(O) \rightarrow \mathbf{Mod}(O_2)$; and (3) O is consistent.

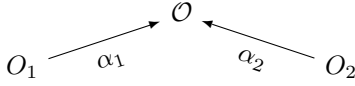


Figure 1: Integration into reference ontology

Example 5. (From Schorlemmer & Kalfoglou 2008, abridged) Suppose that O_1 is a relational scheme. It contains `author_of(person,paper)` and `person(id,name)` with a relationship from `person` to `id`. O_2 is a description logic theory. It contains `Article` $\sqsubseteq \exists \text{author}.\top \sqcap \exists \text{title}.\top$, etc.

The reference ontology \mathcal{O} is a first-order theory. It contains, among others:

$$\begin{aligned} \forall x.(Working_Person(x) &\rightarrow (Thing(x) \wedge \\ &\exists y.(String(y) \wedge Name(x,y)))) \\ \forall x.(Researcher(x) &\rightarrow Working_Person(x)) \end{aligned}$$

Theory interpretations α_1, α_2 can be given as follows:

$$\begin{aligned} \alpha_1(\text{person}(p,n)) &= Researcher(p) \wedge String(n) \wedge \\ &Name(p,n) \\ \alpha_1(\text{author_of}(p,a)) &= Researcher(p) \wedge Article(a) \wedge \\ &Author(a,p) \wedge \exists j.(Journal(j) \\ &\wedge Has_Article(j,a)) \\ \alpha_2(Article(x)) &= Paper(x) \end{aligned}$$

◇

We see a number of problems with this approach (and therefore we will reformulate the example as a heterogeneous refinement on Page 7). First, allowing for arbitrary sentence maps α_i is simply too liberal. For example, α_i could map every sentence to *true*.⁹ It seems more reasonable to use *signature morphisms* and their induced sentence translation maps instead. This approach, however, is less flexible in one aspect: with the approach of (Schorlemmer & Kalfoglou 2008) (using first-order logic), a predicate symbol p may be mapped to a formula φ . However, this is usually better captured by allowing for *derived signature morphisms* (see (Sannella & Burstall 1983)), which here are just signature morphisms into a conservative extension (e.g. an extension by the definition $p(x) \Leftrightarrow \varphi$). Secondly, and more importantly perhaps, there may simply be no suitable common reference ontology at hand. Rather, the common super-ontology should be suitably *constructed* from O_1 and O_2 , identifying certain concepts, while keeping others distinct, leading to the concept of *V-alignment* discussed in the next section. Another possibility is to compare the two ontologies using a view, leading to the concept of *refinement* discussed on Page 6.

4 Heterogeneous Connection

Intuitively, the difference between ‘integrations’ and ‘connections’ is that in the former we combine two ontologies O_1 and O_2 using a typically large and previously-known reference ontology \mathcal{O} . The models of \mathcal{O} are typically much richer than those of O_1 and O_2 . By contrast, *connection* of two ontologies is done in such a way that the respective theories, signatures, and models are kept disjoint, and a (usually small and flexible) *bridge theory* formulated (in a bridge language) over a signature that *goes across* the sort

⁹(Schorlemmer & Kalfoglou 2008) suggest to solve this problem by a possible restriction to conservative translations; however, even then the translation mapping every theorem in O_i to *true* and every non-theorem to *false* still is a valid but useless example.

structure of the components is used to *link together* the two ontologies. Using the general approach of colimits, an overall connection ontology can be *automatically computed* from the bridge theory and the involved ontologies. Moreover, the models of this overall ontology are obtained as amalgamations of models of the individual ontologies — no new structure is added (except from new definitions, which however can always be interpreted uniquely).

4.1 Connection through Alignments

4.1.1 V-Alignments

(Zimmermann et al. 2006) address the problem of alignment without a common reference ontology. Given ontologies O_1 and O_2 , an **interface** (for O_1, O_2)

$$\langle \Sigma, \sigma_1: \Sigma \rightarrow \text{Sig}(O_1), \sigma_2: \Sigma \rightarrow \text{Sig}(O_2) \rangle$$

specifies that (using informal but suggestive notation)

- concepts $\sigma_1(c)$ in O_1 and $\sigma_2(c)$ in O_2 are identified for each concept c in Σ , regardless of whether the concepts have the same name or not, and
- concepts in $O_1 \setminus \sigma(\Sigma_1)$ and $O_2 \setminus \sigma(\Sigma_2)$ are kept distinct, again regardless of whether they have the same name or not.

The resulting common ontology \mathcal{O} is not given a priori, but rather it is computed from the aligned ontologies via the interface. This computation is a pushout in the sense of category theory, which in this case is just a disjoint union with identification of specific parts (namely those given through $\langle \Sigma, \sigma_1, \sigma_2 \rangle$).

V-alignments can thus deal with basic alignment problems, such as *synonymy* (identifying different symbols with the same meaning) and *homonymy* (separating (accidentally) identical symbols with different meaning)—see Figure 2.

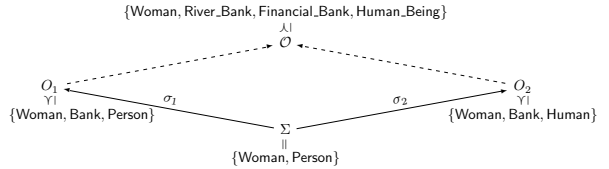


Figure 2: V-alignment: integration through interface (dashed arrows mean definition links automatically computed via colimits)

Note that alignments are encoded entirely into the interface; *finding* an appropriate alignment (i.e. an interface $\langle \Sigma, \sigma_1, \sigma_2 \rangle$) is a different problem, compare (Euzenat & Shvaiko 2007), and independent from the chosen formal representation.

Example 6. In Figure 2, the interface $\langle \Sigma, \sigma_1, \sigma_2 \rangle$ specifies that the two instances of the concept `Woman` as well as `Person` and `Human` are to be identified. This yields two concepts `Woman` and `Human_Being` in the push-out ontology \mathcal{O} obtained along the dashed arrows. Because `Bank` does not appear in the interface, it also determines that the two instances of `Bank` are to be understood as homonyms, and thus generates two new distinct concepts. ◇

However, notion such as *polysemy* are typically understood to relate terms that have a different, but *related* meaning, and can thus not be dealt with by simply identifying symbols or keeping them apart. This problem can be solved, however, by considering

\mathcal{E} -connections a general form of alignment (see (Kutz et al. 2008)). Similarly, (Zimmermann et al. 2006) themselves raise the criticism that V-Alignments do not cover the case where a concept *Woman* in O_1 is aligned with a concept *Person* in O_2 : here, the resulting ontology should turn *Woman* into a subconcept of *Person*. This is not directly possible with the pushout approach.

4.1.2 W-Alignments

In order to solve this problem of V-Alignments, (Zimmermann et al. 2006) introduce W-Alignments. They consist of two V-Alignments, using an intermediate bridge ontology B . The latter can be used to specify subconcept relationships like *Woman* \sqsubseteq *Person* as mentioned above.

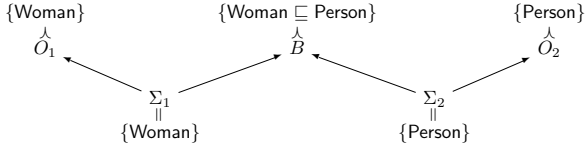


Figure 3: W-alignment: integration through bridge ontology

(Zimmermann et al. 2006) list the behaviour of compositions as a weak point of this approach. However, we see as the main weak point the rather loose coupling of O_1 and O_2 ; indeed, the bridge ontology is something like a super-ontology of a sub-ontology and hence can be anything. Nevertheless, an advantage of W-alignments over semantic integrations into a reference ontology remains: it is not possible to map sentences into the integration ontology in a completely arbitrary fashion. In (Kutz et al. 2008), we have shown that various kinds of alignments can be analysed as certain ‘shapes’ of diagrams that can be represented and reasoned with in HETS.

4.2 Connection through Interface and Colimit

The general idea of combination through an interface by computing a colimit is shown in Fig 4. Here, Σ_1

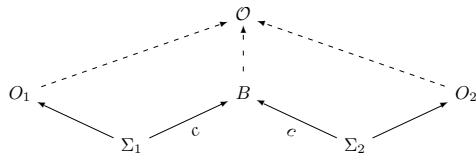


Figure 4: Connection through interface and colimit

is a subsignature of ontology O_1 , Σ_2 a subsignature of ontology O_2 , B an interface formalised in a bridge logic such as $\mathbf{FOL}^{ms=}$, and O the colimit ontology computed from the diagram.

Example 5 can be reformulated in this setting by taking O_1 to be the relational scheme formalisation, O_2 the DL knowledge base, and B the necessary first-order axioms to achieve the desired reconciliation. The ‘reference ontology’ is now obtained as a pushout. The complete specification for this scenario is given in Appendix B.2.

4.3 \mathcal{E} -Connections

Heterogeneous knowledge representation was a major motivation also for the design of ‘modular ontology languages’, such as DDLs (Borgida & Serafini

2003) and \mathcal{E} -connections (Kutz et al. 2002, 2004). We here concentrate on the latter. \mathcal{E} -connections were originally conceived as a versatile and computationally well-behaved technique for combining logics, but were subsequently quickly adopted as a framework for the combination of ontologies in the Semantic Web (Cuenca Grau, Parsia & Sirin 2008).

We here show how the combination of ontologies via such modular languages can be re-formulated as structured heterogeneous ontologies, and indicate how this idea can be generalised to the institutional level.

The general idea behind this combination method is that the interpretation domains of the connected logics are interpreted by disjoint (or sorted) vocabulary and interconnected by means of *link relations*. The language of the \mathcal{E} -connection is then the union of the original languages enriched with operators capable of talking about the link relations.

\mathcal{E} -connections, just as DLs, offer an appealing compromise between expressive power and computational complexity: although powerful enough to express many interesting concepts, the coupling between the combined logics is sufficiently loose for proving general results about the transfer of decidability: if the connected logics are decidable, then their connection will also be decidable.

For lack of space, we can only roughly sketch the formal definitions, and refer the reader to (Kutz et al. 2004) for details.

We assume that the languages \mathcal{L}_1 and \mathcal{L}_2 of two ontologies O_1 and O_2 are pairwise disjoint. To form a connection $\mathcal{C}^{\mathcal{E}}(O_1, O_2)$, fix a non-empty set $\mathcal{E} = \{E_j \mid j \in J\}$ of binary relation symbols. The **basic \mathcal{E} -connection language** is then defined inductively by enriching the respective languages with the basic \mathcal{E} -connection operators $\langle E_j \rangle^1, \langle E_j \rangle^2$, interpreting the link relations.

Fig. 5 displays the connection of two ontologies, by means of a single link relation E . Here, the concept $\langle E \rangle^1(\{a\})$ of O_1 ‘corresponds’ to the nominal $\{a\}$ of ontology O_2 : it collects the set of all those points in O_1 that ‘can be seen’ from a (in O_2) along the relation E .

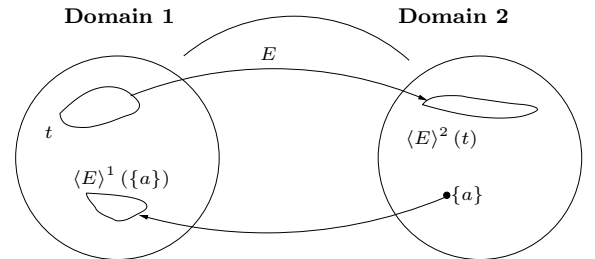


Figure 5: A two-dimensional connection.

Formally, the semantics is as follows: a structure

$$\mathfrak{M} = \langle \mathfrak{W}_1, \mathfrak{W}_2, \mathcal{E}^{\mathfrak{M}} = (E_j^{\mathfrak{M}})_{j \in J} \rangle,$$

where $\mathfrak{W}_i = (W_i, \cdot^{\mathfrak{M}_i})$ is an interpretation of O_i for $i \in \{1, 2\}$ and $E_j^{\mathfrak{M}} \subseteq W_1 \times W_2$ for each $j \in J$, is called an **interpretation** for $\mathcal{C}^{\mathcal{E}}(O_1, O_2)$. Given concepts C_i of ontology O_i , for $i = 1, 2$, denoting subsets of W_i , the semantics of the basic \mathcal{E} -connection operators is

$$\begin{aligned} \langle E_j \rangle^1 C_2^{\mathfrak{M}} &= \{x \in W_1 \mid \exists y \in C_2^{\mathfrak{M}} (x, y) \in E_j^{\mathfrak{M}}\} \\ \langle E_j \rangle^2 C_1^{\mathfrak{M}} &= \{x \in W_2 \mid \exists y \in C_1^{\mathfrak{M}} (x, y) \in E_j^{\mathfrak{M}}\} \end{aligned}$$

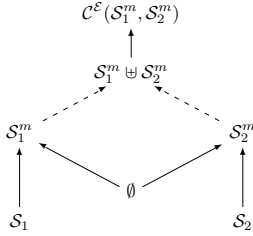


Figure 6: \mathcal{E} -connections as a structured heterogeneous theory

\mathcal{E} -connections can be considered as many-sorted heterogeneous theories: component ontologies can be formulated in different logics, but have to be build 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) (Borgida & Serafini 2003) and various \mathcal{E} -connections now lies in the expressivity of the ‘link language’ \mathcal{L} connecting the different ontologies. While the 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{ALCT}^{ms} .

Such many-sorted theories can easily be represented in a diagram as shown in Fig. 6. Here, we first (conservatively) obtain a disjoint union $S_1^m \uplus S_2^m$ as a pushout, where the component ontologies have been turned into sorted variants (using an institution comorphism from the single-sorted to the many-sorted logic), and the empty interface guarantees that no symbols are shared at this point. An \mathcal{E} -connection KB in language $\mathcal{C}^E(S_1^m, S_2^m)$ is then obtained as a (typically not conservative) theory extension.

The idea to ‘connect’ logics can be elegantly generalised to the institutional level (compare Baader & Ghilardi (2007) who note that their ‘connections’ are an instance of a more general co-comma construction), but details have to remain sketchy here. However, it should be clear from the above that our Grothendieck institution approach is general enough to formally capture such connections.

Note that this generalises the \mathcal{E} -connections of (Kutz et al. 2002), the DDLs of (Borgida & Serafini 2003), as well as the connections of Baader & Ghilardi (2007) 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.

While there are no implemented, specialised algorithms available deciding satisfiability in \mathcal{E} -connections (except limited support in the Pellet system (Cuenca Grau, Parsia & Sirin 2008) which is currently being added to HETS as a supported prover), semi-decidable reasoning for more expressive \mathcal{E} -connections is provided by HETS through suitable translation by a comorphisms in a supported logic.

5 Heterogeneous Refinements

Integrations and connections are essentially *symmetric* combination techniques in the sense that the order in which component ontologies participate in the overall combination is irrelevant. Rather, the difference lies in the way ‘local’ signatures are mapped into the overall signature. In contrast to this a heterogeneous refinement is a asymmetric technique, stating that all

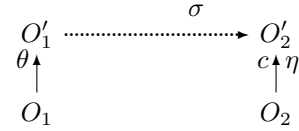


Figure 7: A refinement Diagram.

axioms of a ‘coarser’ ontology are also true in a ‘finer’ (refined) one.

Definition 5. Given two ontologies O_1 and O_2 in the same logic, O_2 is called a **refinement** of O_1 if there is a theorem link $O_1 \xrightarrow{\sigma} O_2$ that follows from the underlying development graph.

Now let ontologies O_1 and O_2 in logics \mathcal{L}_{O_1} and \mathcal{L}_{O_2} be given, such that there is a logic \mathcal{L} with comorphisms $\mathcal{L}_{O_1} \xrightarrow{\theta} \mathcal{L}$ and $\mathcal{L}_{O_2} \xrightarrow{\eta} \mathcal{L}$, where η is model-theoretically conservative.

The translations of the ontologies along the comorphism are referred to as O_1' and O_2' . We call O_2 a **heterogeneous refinement** of O_1 if there is a theorem link $O_1' \xrightarrow{\sigma} O_2'$ that follows from the underlying development graph. \diamond

Proposition 1. For a heterogeneous refinement, any O_2 -model can be translated to an O_1 -model, and moreover, logical consequence is preserved along refinement: for $\sigma(\theta(\varphi)) = \eta(\psi)$, $O_1 \models \varphi$ implies $O_2 \models \eta(\psi)$.

A heterogeneous refinement as given in Def. 5 is depicted in Fig. 7 (here, c denotes conservativity). In HETCASL, this concept is addressed by the notion of **view** creating a proof obligation, as discussed in more detail in Appendix. A.

The notion of a heterogeneous refinement also leads to the definition of heterogeneous sub-ontology.

Definition 6. We call an ontology O_1 a (*heterogeneous*) **sub-ontology** of O_2 iff O_2 is a (heterogeneous) refinement of O_1 .

The standard notion of sub-ontology as a subset of the axioms (Haase et al. 2005, Kalyanpur et al. 2007) is recovered in the homogeneous case if σ additionally is an injection.

Moreover, our abstract definition of refinement entails the common definition in software engineering: consider O_1 to be a specification of a program in an algebraic specification language, and O_2 its implementation in a programming language. Refinements are a common problem in the world of ontologies as well: establish whether a domain ontology is consistent with respect to the knowledge represented in a foundational ontology. Consider a domain ontology written in \mathcal{OWL} -DL that is expected to refine the abstract knowledge given in DOLCE, written in **FOL**. In this case, the domain ontology should be a heterogeneous refinement of DOLCE (or of a part of DOLCE, if one considers hiding).

5.1 From Relational Scheme to Ontology

Recall Example 5 of a semantic integration into a reference ontology. The kind of integration required here can be dealt with much more elegantly as a heterogeneous refinement. Consider the HETCASL specification given in Fig. 9. Here, **Biblio_DL** is a DL ontology about bibliographical information, written in a concrete syntax close to Manchester Syntax (Horridge et al. 2006), and **Biblio_RS** is the scheme of a

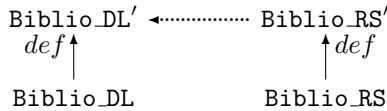


Figure 8: A view from RELSCHEME to HETDL.



Figure 10: A view from HETDL to RELSCHEME.

relational database intended to capture similar knowledge. Assume we want to show that the ontology is a refinement of the database schema, as illustrated in Fig. 8. A view `Biblio_RS_in_DL` is used for this pur-

```

logic DL
spec Biblio_DL =
  Class: Researcher
  SubclassOf: name some Thing
  Class: Article
  SubclassOf: author some Thing, title some Thing
  Class: Journal
  SubclassOf: name some Thing, hasArticle some Thing,
              impactFactor some Thing
end

logic RelScheme
spec Biblio_RS =
  Tables
  person(key id:integer, name:string)
  author_of(person, paper:integer)
  paper(key id:integer,title:string,
         published_in:integer)
  journal(key id:integer,name:string,
           impact_factor:float)
  Relationships
  author_of[person]    -> person[id]  one_to_many
  author_of[paper]    -> paper[id]   one_to_many
  paper[published_in] -> journal[id] one_to_many
end

logic CASL
view Biblio_RS_in_DL : Biblio_RS to
{ Biblio_DL with logic DL -> CASL
  then %def
    preds
    journal(j,n,f:Thing) <=>
      Journal(j) /\ name(j,n) /\ impactFactor(j,f);
    paper(a,t,j:Thing) <=>
      Article(a) /\ Journal(j) /\ hasArticle(j,a) /\
      title(a,t);
    author_of(p,a:Thing) <=>
      Researcher(p) /\ Article(a) /\ author(p,a);
    person(p,n:Thing) <=> Researcher(p) /\ name(p,n)
  } = logic RelationalScheme -> CASL
end

```

Figure 9: Heterogeneous specification in HETCASL.

pose, stating that the ontology satisfies the relational scheme axioms (referential integrity constraints). Of course, this is not possible literally, but rather the ontology is mapped to first-order logic (CASL) and then definitionally (look at the `%def`) extended to `Biblio_DL'` with a definition of the database tables in terms of the ontology classes and properties. Also, `Biblio_RS` is translated to first-order logic, yielding `Biblio_RS'`, and the view expresses a theory morphism from `Biblio_RS'` to `Biblio_DL'`.

The involved signature and theory morphisms live in the Grothendieck institution. Thus, we can avoid the use of arbitrary maps α_i as in (Schorlemmer & Kalfoglou 2008), and instead rely entirely on (Grothendieck) signature morphisms. Actually, the above view is *not* provable—but it is if an inverse of the role `hasArticle` is introduced and used to restrict the class `Article`.

5.2 From Ontology to Relational Scheme

Dually, assume we are given an ontology that is supposed to logically describe a database scheme.

We again use `Biblio_RS` and `Biblio_DL`, as given in Fig. 9. We now have a heterogeneous refinement from `Biblio_DL` to `Biblio_RS`, as depicted in Fig. 10. The rest of the discussion is analogous to the previous section, and hence left out. The specification of this refinement containing the integrity constraints is given in Appendix B.1.

6 Outlook and Future Work

We have introduced an abstract framework for the study of structured heterogeneous ontologies, allowing for a systematic analysis of conceptual and algorithmic problems in heterogeneous environments that were previously considered rather disparate. In particular, we have analysed in detail a specific example heterogeneous ontology from the literature, consisting of a bibliographical database and a related ontology. We have shown that these ontologies can be heterogeneously combined in different ways: (1) via *integration* in a common reference ontology, which is known beforehand, (2) via *connection* through a bridge theory, which operates directly on the involved ontologies and allows for the *automatic* construction of a combined ontology via colimits, and (3) via *refinement*, which provides the strongest relation between the ontologies: namely, that each bibliographical database satisfying the given relational scheme's integrity constraints also gives rise to a model of the ontology, and vice versa. It is not surprising that this strong combination via refinement was only possible after extending the ontology in a suitable way, leading to a better matching between the relational scheme and the ontology.

We believe that these general patterns of ontology combination can also be found in other examples, also involving completely different application domains (and not necessarily being connected with databases). Indeed, the structured reasoning support that our approach allows has already been used to answer questions that 'standard' automated reasoning can not tackle: the consistency of the first-order version of the foundational ontology DOLCE (reformulated as a HETCASL specification) can be verified by model-checking a view into a finite specification of a model for DOLCE.

Currently, we are working on integrating a tool for the discovery of theory morphisms into the Heterogeneous Tool Set, which would allow (semi)-automatic structuring of ontologies.

Acknowledgements

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

We thank John Bateman, Mihai Codescu, Joana Hois, and Lutz Schröder for fruitful discussions.

References

- Adámek, J., Herrlich, H. & Strecker, G. (1990), *Abstract and Concrete Categories*, Wiley, New York.
- Alagić, S. & Bernstein, P. A. (2002), A Model Theory for Generic Schema Management, in ‘Proc. of DBPL-01’, Vol. 2397 of *LNCS*, Springer, pp. 228–246.
- Baader, F. & Ghilardi, S. (2007), ‘Connecting Many-Sorted Theories’, *The Journal of Symbolic Logic* **72**(2), 535–583.
- Bench-Capon, T. J. M. & Malcolm, G. (1999), Formalising Ontologies and Their Relations, in ‘Proc. of DEXA-99’, Vol. 1677 of *LNCS*, Springer, pp. 250–259.
- Bidoit, M. & Mosses, P. D. (2004), *CASL User Manual*, LNCS Vol. 2900 (IFIP Series), Springer.
- Borgida, A. & Serafini, L. (2003), ‘Distributed Description Logics: Assimilating Information from Peer Sources’, *Journal of Data Semantics* **1**, 153–184.
- Codescu, M. & Mossakowski, T. (2008), Heterogeneous colimits, in F. Boulanger, C. Gaston & P.-Y. Schobbens, eds, ‘MoVaH’08 Workshop on Modeling, Validation and Heterogeneity’.
- CoFI (The Common Framework Initiative) (2004), *CASL Reference Manual*, LNCS Vol. 2960 (IFIP Series), Springer.
- Cuenca Grau, B., Honavar, V., Schlicht, A. & Wolter, F., eds (2007), *2nd International Workshop on Modular Ontologies (WoMO-07)*, Vol. 315, CEUR Workshop Proceedings, (K-CAP) Whistler, BC, Canada.
- Cuenca Grau, B., Horrocks, I., Kazakov, Y. & Sattler, U. (2008), ‘Modular Reuse of Ontologies: Theory and Practice’, *J. of Artificial Intelligence Research (JAIR)* **31**. To appear.
- Cuenca Grau, B., Parsia, B. & Sirin, E. (2008), Ontology Integration Using \mathcal{E} -connections, in H. Stuckenschmidt & S. Spaccapietra, eds, ‘Ontology Modularization’, Springer. To Appear.
- Diaconescu, R. (2002), ‘Grothendieck institutions’, *Applied Categorical Structures* **10**, 383–402.
- Diaconescu, R. (2008), *Institution-independent Model Theory*, Studies in Universal Logic, Birkhäuser.
- Euzenat, J. & Shvaiko, P. (2007), *Ontology Matching*, Springer, Heidelberg.
- Goguen, J. A. (2005), ‘Data, Schema, Ontology and Logic Integration’, *Logic J. of the IGPL* **13**, 685–715.
- Goguen, J. A. (2006), Information Integration in Institutions, in L. Moss, ed., ‘Jon Barwise Memorial Volume’, Indiana University Press. To appear.
- Goguen, J. A. & Burstall, R. M. (1992), ‘Institutions: Abstract Model Theory for Specification and Programming’, *Journal of the ACM* **39**, 95–146.
- Goguen, J. A. & Roşu, G. (2002), ‘Institution morphisms’, *Formal aspects of computing* **13**, 274–307.
- Guerra, S. (2001), ‘Composition of Default Specifications’, *J. Log. Comput.* **11**(4), 559–578.
- Haase, P., Honavar, V., Kutz, O., Sure, Y. & Tamin, A., eds (2006), *1st International Workshop on Modular Ontologies (WoMO-06)*, Vol. 232, CEUR Workshop Proceedings, (ISWC) Athens, Georgia, USA.
- Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. (2005), A framework for handling inconsistency in changing ontologies, in ‘Proc. of the 4th International Semantic Web Conference (ISWC-05)’, Vol. 3729 of *LNCS*, Springer, pp. 353–367.
- Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R. & Wang, H. H. (2006), The Manchester OWL Syntax, in ‘OWL: Experiences and Directions’.
- Horrocks, I., Kutz, O. & Sattler, U. (2006), The Even More Irresistible *SRIOQ*, in ‘Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006)’, AAAI Press, pp. 57–67.
- Kalyanpur, A., Parsia, B., Horridge, M. & Sirin, E. (2007), Finding all Justifications of OWL DL Entailments, in ‘Proc. of ISWC/ASWC2007’, LNCS Vol. 4825, Springer, pp. 267–280.
- Konev, B., Lutz, C., Walther, D. & Wolter, F. (2008), Formal properties of modularization, in H. Stuckenschmidt & S. Spaccapietra, eds, ‘Ontology Modularization’, Springer.
- Kutz, O., Lutz, C., Wolter, F. & Zakharyashev, M. (2004), ‘ \mathcal{E} -Connections of Abstract Description Systems’, *Artificial Intelligence* **156**(1), 1–73.
- Kutz, O. & Mossakowski, T. (2007), Modules in Transition: Conservativity, Composition, and Colimits, in ‘2nd Int. Workshop on Modular Ontologies (WoMO-07)’. K-CAP, Whistler BC, Canada.
- Kutz, O. & Mossakowski, T. (2008), Conservativity in Structured Ontologies, in ‘Proc. of the 18th European Conference on Artificial Intelligence (ECAI-08)’, Patras, Greece. Forthcoming.
- Kutz, O., Mossakowski, T. & Codescu, M. (2008), Shapes of Alignments: Construction, Combination, and Computation, in U. Sattler & A. Tamin, eds, ‘Proc of the 1st Workshop on Ontologies: Reasoning and Modularity (WORM-08)’, CEUR-WS, Vol-348, ESWC, Tenerife, Spain.
- Kutz, O., Wolter, F. & Zakharyashev, M. (2002), Connecting abstract description systems, in ‘Proc. of the 8th Conference on Principles of Knowledge Representation and Reasoning (KR-02)’, Morgan Kaufmann, pp. 215–226.
- Mac Lane, S. (1998), *Categories for the Working Mathematician*, 2nd edn, Springer, Berlin.
- Madhavan, J., Bernstein, P., Domingos, P. & Halevy, A. (2002), Representing and reasoning about mappings between domain models, in ‘Proc. of AAAI 2002’, Edmonton, Canada.
- Mossakowski, T. (2002), Comorphism-based Grothendieck logics, in ‘Mathematical Foundations of Computer Science’, Vol. 2420 of *LNCS*, Springer, pp. 593–604.
- Mossakowski, T., Autexier, S. & Hutter, D. (2006), ‘Development graphs—proof management for structured specifications’, *Journal of Logic and Algebraic Programming* **67**(1–2), 114–145.

Mossakowski, T., Maeder, C. & Lüttich, K. (2007a), The Heterogeneous Tool Set, in O. Grumberg & M. Huth, eds, ‘TACAS 2007’, Vol. 4424 of *LNCS*, Springer, pp. 519–522.

Mossakowski, T., Maeder, C. & Lüttich, K. (2007b), The Heterogeneous Tool Set, in B. Beckert, ed., ‘VERIFY 2007’, Vol. 259, CEUR-WS.

Sannella, D. & Burstall, R. (1983), Structured theories in LCF, in ‘Proc. 8th Colloq. on Trees in Algebra and Programming’, Vol. 159 of *LNCS*, Springer, pp. 377–391.

Sattler, U. & Tamilin, A., eds (2008), *Workshop on Ontologies: Reasoning and Modularity (WORM-08)*, Vol. 348, CEUR Workshop Proceedings, ESWC, Tenerife, Spain.

Schorlemmer, M. & Kalfoglou, Y. (2008), ‘Institutionalising Ontology-Based Semantic Integration’, *Journal of Applied Ontology*. . To appear.

Wöfl, S., Mossakowski, T. & Schröder, L. (2007), Qualitative constraint calculi: Heterogeneous verification of composition tables, in ‘Proc. FLAIRS 2007’, pp. 665–670.

Zimmermann, A., Krötzsch, M., Euzenat, J. & Hitzler, P. (2006), Formalizing Ontology Alignment and its Operations with Category Theory, in ‘Proc. of FOIS-06’, pp. 277–288.

Appendix

A Heterogeneous Reasoning with HETS

A heterogeneous proof calculus based on development graphs, as they have been defined earlier in this paper, is implemented in the Heterogeneous Tool Set HETS (see (Mossakowski et al. 2007a,b)). It supports a multitude of logics, given as institutions, that can be used in formal specification. Examples are many-sorted first-order logic \mathbf{FOL}^{ms} with equality underlying CASL and the description logic $\mathbf{SR\text{OIQ}(D)}$ (and its sublogics \mathcal{EL} and \mathcal{ALC}) underlying \mathbf{OWL} 1.1. Several concrete syntaxes for \mathbf{OWL} are supported, including Manchester Syntax (Horridge et al. 2006) and its extension to $\mathbf{SR\text{OIQ}(D)}$. Moreover, relational schemes, as well as $\mathbf{QS5}$ (using syntax of the CASL language) are supported.

Between many of the logics, comorphisms are defined and implemented. For most of them, a comorphism into CASL exists, allowing heterogeneous specifications. A heterogeneous proof calculus (see (CoFI (The Common Framework Initiative) 2004)) for development graphs shifting proof obligations between nodes into nodes is implemented, too.

To prove obligations of a node (= in a single logic), several (first-order) \mathbf{FOL} (SPASS, Darwin) and (higher-order) \mathbf{HOL} (Isabelle) provers are integrated into HETS, enabling proof support for heterogeneously specified ontologies. The DL reasoner Pellet is supported as a consistency checker for ontologies.

The development graph calculus integrated into HETS can be used for all the integration, connection, and refinement techniques for ontologies discussed in this paper: the verification of semantic integrations as well as refinements are directly supported. In the case colimit computation is needed, as in W -alignments, this can be calculated via HETS’ built-in colimit feature described in (Codescu & Mossakowski 2008), that also allows the ‘approximation’ of colimits.

A specific example for such a heterogeneous, multi-level reasoning is provided by the verification of the RCC composition tables (Wöfl et al. 2007). Here, the majority of proof obligations can be resolved by various, fully automatic reasoners. However, a proof obligation requiring the second-order theory of real numbers needs an interactive proof in the Isabelle prover.

B Heterogeneous Specifications in HETCASL

In the following, we present additional specifications for the interested reader.

B.1 Specification: From Ontology to Relational Scheme

We give the specification for Section 5.2. The specification $\mathbf{Biblio_DL}$ is written in HETDL, a concrete Syntax for $\mathbf{SR\text{OIQ}(D)}$, that is very closely related to the Manchester Syntax (Horridge et al. 2006). Our language is almost self-explanatory, and follows the naming conventions of \mathbf{OWL} .

E.g., the stanza

```
Class: Researcher
SubclassOf: name some string
```

defines a class $\mathbf{Researcher}$ that is a subclass of all \mathbf{Things} having a \mathbf{name} of type \mathbf{string} .

$\mathbf{Biblio_RS}$ is written in $\mathbf{RELScheme}$, see Example 2. Following the keyword \mathbf{Tables} , the signature of a specification is defined. It consists of a set of tables, each having columns of a particular data type. In $\mathbf{person}(\mathbf{key\ id:} \mathbf{pointer, name:string})$, a table with the name \mathbf{person} is defined, having the columns \mathbf{id} and \mathbf{name} with \mathbf{id} being the key of this table. If \mathbf{key} occurs more than once in a table, we have a compound key. The sentences following the keyword $\mathbf{Relationships}$ define relations between different tables. The sentence $\mathbf{author_of[person] \rightarrow person[id] one_to_many}$ means that the column \mathbf{person} of the table $\mathbf{author_of}$ is in $\mathbf{one_to_many}$ relationship with the column \mathbf{id} of \mathbf{person} .

The view $\mathbf{Biblio_DL_in_RS}$ is written in CASL. The link between $\mathbf{Biblio_DL}$ and $\mathbf{Biblio_RS}$ is established via several CASL sentences. Please note that the view here goes into the opposite direction compared to the one in Fig. 9.

```
logic DL
spec Biblio_DL =
  Class: Researcher
  SubclassOf: name some string

  ObjectProperty: hasArticle
  InverseOf: hasJournal

  Class: Article
  SubclassOf: author some Thing, title some string,
             hasJournal some Journal

  Class: Journal
  SubclassOf: name some string,
             hasArticle some Thing,
             impactFactor some integer
end

logic RelScheme
spec Biblio_RS =
  Tables
  person(key id:pointer, name:string)
  author_of(person, paper:pointer)
  paper(key id:pointer, title:string,
        published_in:pointer)
  journal(key id:pointer, name:string,
          impact_factor:integer)

  Relationships
  author_of[person]      -> person[id] one_to_many
```

```

author_of[paper]    -> paper[id]    one_to_many
paper[published_in] -> journal[id]  one_to_many
end

logic CASL
view Biblio_DL_in_RS : Biblio_DL to
{ Biblio_RS with logic RelScheme -> CASL
  then %def
    preds Researcher(x:pointer) <=>
      (exists n:string; a:pointer.
        person(x,n) /\ author_of(x,a));
    Article(x:pointer) <=>
      (exists t:string; j:integer.paper(x,t,j));
    Journal(x:pointer) <=>
      (exists n:string; i:float.journal(x,n,i));
    name(x:pointer;n:string) <=>
      person(x,n);
    hasArticle(x,j:pointer) <=>
      (exists n,t:string; i:integer .
        journal(x,n,i) /\ paper(j,t,x));
    hasJournal(j,x:pointer) <=>
      (exists n,t:string; i:integer .
        journal(x,n,i) /\ paper(j,t,x));
    author(a,p:pointer) <=>
      (exists t,n:string; p:pointer .
        paper(a,t,p) /\ person(p,n));
    title(a:pointer; t:string) <=>
      (exists p:pointer . paper(a,t,p));
    impact_factor(j:pointer;f:integer) <=>
      (exists n:string . journal(j,n,f));
  }
end

```

Such a specification can be parsed and analysed by the tool HETS, which displays it as a development graph as in Fig. 11.

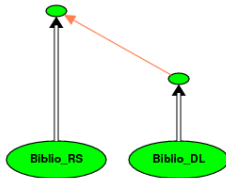


Figure 11: Development graph in HETS.

In this picture, red arrows are theorem links, and outlined thick black arrows are heterogeneous definition links. The picture shows the situation shown in the diagram of Fig. 10.

To prove this view, the development graph calculus has to be applied first. Then a prover needs to be invoked on the upper left node, that by now has become red: a node with proof obligations. The proof obligations can e.g. be proved with the SPASS theorem prover.

B.2 Specification: Integration through Interface and Colimit

In this section, an example specification for a W-alignment is given.

```

logic DL
spec Biblio_DL =
  Biblio_DL_Sign
then
  Class: Researcher
  SubclassOf: name some string

  ObjectProperty: hasArticle
  InverseOf: hasJournal

  Class: Article
  SubclassOf: author some Thing, title some string,
              hasJournal some Journal

  Class: Journal
  SubclassOf: name some string,
              hasArticle some Thing,
              impactFactor some integer
end

```

```

spec Biblio_DL_Sign =
  Class: Researcher
  DataProperty: name

  Class: Article
  ObjectProperty: author
  DataProperty: title
  ObjectProperty: hasJournal

  Class: Journal
  ObjectProperty: hasArticle
  DataProperty: impactFactor
end

logic RelScheme
spec Biblio_RS_Sign =

  Tables
  person(key id:pointer, name:string)
  author_of(person, paper:pointer)
  paper(key id:pointer,title:string,
         published_in:pointer)
  journal(key id:pointer,name:string,
          impact_factor:integer)
end

spec Biblio_RS =
  Biblio_RS_Sign
then
  Relationships
  author_of[person]    -> person[id]    one_to_many
  author_of[paper]    -> paper[id]    one_to_many
  paper[published_in] -> journal[id]  one_to_many
end

logic CASL
spec Interface =
{Biblio_RS_Sign with logic RelScheme -> CASL}
and
{Biblio_DL_Sign with logic DL -> CASL
  with Thing |-> pointer}
then
  forall a,j,p,x:pointer;n,t:string;f:integer
  . journal(j,n,f) <=> Journal(j) /\ name(j,n)
    /\ impactFactor(j,f)
  . paper(a,t,j) <=> Article(a) /\ Journal(j)
    /\ hasArticle(j,a) /\ title(a,t)
  . author_of(p,a) <=> Researcher(p) /\ Article(a)
    /\ author(p,a)
  . person(p,n) <=> Researcher(p) /\ name(p,n)
  . Researcher(x) <=> (exists q:pointer;m:string .
    person(x,m) /\ author(x,q))
  . Article(x) <=> (exists q:pointer;m:string .
    paper(x,m,q))
  . Journal(x) <=> (exists m:string;i:integer .
    journal(x,n,i))
end

```

The picture in figure 12 shows the development

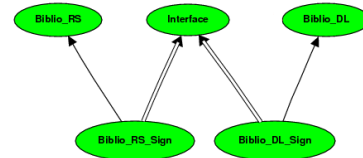


Figure 12: W-alignment in HETS.

graph of the above specification in HETS. In this figure, the outlined thick black arrows are heterogeneous definition links, while the normal black arrows are definition links. The picture shows the situation depicted in the diagram of Fig. 3. Its colimit can be automatically computed by HETS.