Chinese Whispers and Connected Alignments

Oliver Kutz[1], Immanuel Normann[2], Till Mossakowski[3], and Dirk Walther[4]

[1] Research Center on Spatial Cognition (SFB/TR 8), University of Bremen, Germany
okutz@informatik.uni-bremen.de
[2] Department of Linguistics and Literature, University of Bremen, Germany
normann@uni-bremen.de
[3] DFKI GmbH and SFB/TR 8 Spatial Cognition, University of Bremen, Germany
Till.Mossakowski@dfki.de
[4] Faculty of Informatics, Technical University of Madrid (UPM), Spain
dwalther@fi.upm.es

Abstract. This paper investigates the idea to treat repositories of ontologies as interlinked networks of ontologies, formally captured by the notion of a hyperontology. We apply standard matching algorithms to automatically create the linkage structure of the repository by performing pairwise matching. Subsequently, we define a modular workflow to construct compositions of alignments for any finite number of ontologies. This workflow employs and makes interoperable several tools from the ontology engineering world, comprising matching, reasoning, and structuring tools. This allows for an automatic construction of 'ontological synsets', visualisation of the linkage structure, modular ontology extraction based on alignment, and a study and empirical analysis of consistency propagation (the Chinese Whispers problem).

Keywords: Hyperontologies; Connected Alignments; Modularity; Consistency

1 Introduction and Motivation

Ontology matching and alignment based on statistical methods is a relatively developed field, with yearly competitions since 2004 comparing the various strengths and weaknesses of existing algorithms.[5] Such approaches can be quite successful, but ignoring the logical content of an ontology can yield at least three severely negative outcomes:[6] (1) concepts might be matched because of accidental naming similarity whilst their logical meaning in the two ontologies might be completely different; (2) different naming conventions might not allow for matchings based on surface similarity, whilst large parts of two axiomatised ontologies might be largely identical, or equivalent; (3) an alignment based on purely empirical measures, although establishing some meaningful connections, is often inconsistent overall.

OK: harmonise terminology throughout!!!

In this paper we aim at exploring the degrees to which statistical alignment may lead to inconsistency in the merged ontologies. More precisely, we aim at investigating the

[5] See http://oaei.ontologymatching.org/2009/
[6] The lack of semantics involved in the evaluation of such alignments has been clearly articulated already in [3,2].

effects, both theoretically and practically, of **connected matching**, i.e. aligning several ontologies that match (non-trivially) pairwise. Our general approach is to treat large repositories of ontologies (in the order of hundreds of ontologies) as our starting point to perform pairwise matching in order to obtain an interlinked network of ontologies. Formally, such networks are captured by the notion of a **hyperontology** [12].

Our work in progress is intended to answer questions such as the following:

> *If pairwise alignments are still consistent, how and when can we align further ontologies (in various orders) before we drift into inconsistency? Is there a significant difference between matching thematically closely related ontologies (as determined by a human) and randomly selected ones? How and when can we reduce the question of consistency of aligned ontologies to the consistency of the aligned sub-ontologies (i.e. modules generated by the matched sub-signatures)?*

In this paper, we set up the theoretical background and neccessary engineering environment to give meaningful answers to such questions, and study the effects of matching ontologies in different order by looking at some specific examples. In our related paper [16], we have studied techniques of information hiding to allow a user to explore the complex structure resulting from pairwise matching on large sets of ontologies. We here focus on the interoperability problem between matching, modularity, and structuring tools, and the theoretical problem of reducing the consistency of sets of aligned ontologies to sets of (corresponding) aligned modules.

2 Hyperontologies and Structuring

2.1 Structured Ontologies and Repositories as Hyperontologies

Structured Ontologies develop a rather abstract view of heterogeneously structured ontology, encompassing essentially all logics being used in ontology design today and allowing for the modelling of the most complex relationships between ontologies. Technically, we have formalised several logics that are important from an ontology design perspective as so-called *institutions* [4,5], including the description logic $\mathcal{SROIQ}(D)$ and various variants of first-order logic and different modal logics, and supply *institution comorphisms* (logic translations) as mappings between them.

Classification of Combination Techniques systematise the field of 'combining ontologies' by identifying three classes of such combinations: *refinements*, *integrations*, and *connections*. The differentiating criteria are the use of signatures in the overall combination and the corresponding model-theoretic properties.

Hyperontologies introduce a notion of heterogeneously structured ontology, which also affords distributed networks of ontologies written in different formalisms, which we call *hyperontologies*.

Reasoning with Combinations analyse how various well-known ontology design and combination techniques fit into these abstract categories, including structuring through

conservative extensions, ontology alignments,[7] $\mathcal{E}$-connections, and database-scheme–ontology reconciliation.

Tool Support discuss how the tool HETS (Heterogeneous Tool Set) can support various reasoning and ontology engineering tasks and indicate the current and planned tool support for existing ontology languages and reasoners.

The main features of our approach to ontology design may then be summarised as follows:

- The ontology designer can use description logics to specify most parts of an ontology, and can use first-order (or even higher-order) logic where needed. Moreover, the overall ontology can be assembled from (and can be split up into) semantically meaningful parts ('modules') that are systematically related by structuring mechanisms. These parts can then be re-used and/or extended in different settings.
- Institution theory provides logic translations between different ontology languages, translating the syntax and semantics of different formalisms.
- Various concepts of 'ontological module' are covered, including simple imports (extensions) and union of theories, as well as conservative and definitional extensions. We here consider conservative extensions as a way of ('a priori') structuring ontologies, rather than as a methodology to ('a posteriori') cutting large ontologies into pieces which lie conservatively within the whole. However, we consider the latter (algorithmic) approach as assistive to, for instance, verifying a desired conservative design.
- Structuring into modules is made explicit in the ontology and generates so-called proof obligations, e.g. for conservativity. Proof obligations can also be used to keep track of desired consequences of an ontology (module), especially during the design process.
- Re-using (parts of) ontologies whilst renaming (parts of) the signature is handled by *symbol maps* and *hiding symbols*: essentially, this allows the internalisation of (strict) alignment mappings.
- The approach allows heterogeneous refinements: it is possible to prove that an ontology O_2 is a refinement of another ontology O_1, formalised in a different logic. For instance, one can check if a domain ontology is a refinement of (a part of) a foundational one. There are two interesting by-products of the definition of heterogeneous refinement: firstly, it provides a rather general definition of heterogeneous sub-ontology, and secondly, it can be used to give a definition of equivalence of ontologies across different ontology languages.
- The tool HETS provides parsing, static analysis and proof management for heterogeneous logical theories. It can visualise the module structure of complex logical theories, using so-called development graphs. For individual nodes (corresponding to logical theories) in such a graph, the concept hierarchy can be displayed. Moreover, HETS is able to prove intended consequences of theories, prove refinements between theories, or demonstrate their consistency, and compute normal forms and colimits

[7] Previously discussed in [11].

(also for heterogeneous specifications). This is done by integrating several first-order provers and model-finders (SPASS, DARWIN), a higher-order prover (ISABELLE), as well as the DL reasoners PELLET and FACT++.

OK: introduce signature morphisms and some other basics here formally.

2.2 Alignment as Colimit Computation

V-Alignments [23] address the problem of alignment without a common reference ontology. Given ontologies O_1 and O_2, an **interface** (for O_1, O_2)

$$\langle \Sigma, \sigma_1 \colon \Sigma \longrightarrow \mathsf{Sig}(O_1), \sigma_2 \colon \Sigma \longrightarrow \mathsf{Sig}(O_2) \rangle$$

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 Fig. 1.

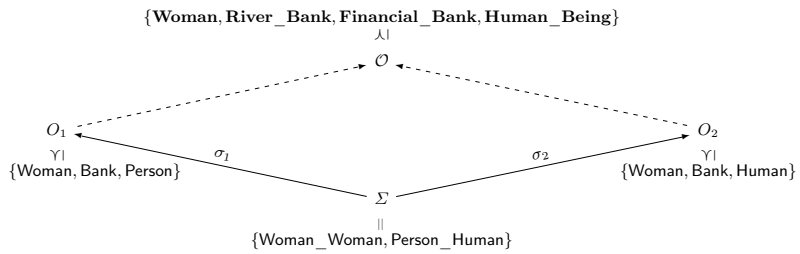

Fig. 1. V-alignment: merge through interface (dashed arrows are automatically computed via colimits)

Example 1. In Fig. 1, 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. 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 as a general form of alignment (see [11]). Similarly, [23] 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.

Someone: describe how the results from a matcher are rewritten into a Spec that codifies a V-alignment. The colimit will compute the result of gluing together.

An important fact for us is that compositions of alignments are associative.

Lemma 1. *Compositions of alignments are associative.*

Proof. This has been shown by Zimmermann *et al.* [23] by observing that the composition of alignments corresponds to spans in category theory, so can be obtained by the use of the categorical construction called pullback. The associativity of compositions can be directly seen from the illustration below taken from [23].

OK: Explain picture properly or make new.

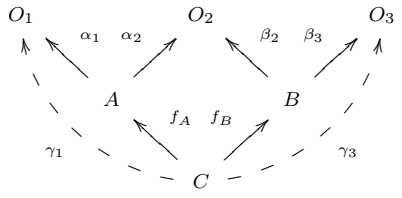

Fig. 2. Composition of alignments is associative

OK: not really chinese whispers because not sequential

This means that the order in which we apply alignments consecutively does not matter, rather, it is important just which subset of pairs of ontologies we pick for (pairwise) matching.

3 Consistency and the Chinese Whispers Effect

IN: nice picture of the hyper

OK: compute the combinatorics of iterative alignments as compared to pairwise matching:

The game of **Chinese Whispers**[8] is played as follows: n persons are arranged in a certain (typically circular) order such that for each person P_i there is a j such that P_i exchanges a message with P_j. The point of the game is to observe the *distortion of the message* as it travelles from P_1 along the communication channel. We here are interested in the effects of playing Chinese Whispers with ontologies, where the pairwise matching replaces the transmission of a message, i.e. the messages being exchanged are of the form: *"O_i and O_j agree that concept C of O_i is synonymous with concept D of O_j"*.

We make the following idealisations concerning 'matching' a) we assume that in pairwise matching the order does not matter, i.e. matching O_1 with O_2 yields the same

[8] In the United States, "Telephon" is the most common name for the game. The name "Chinese whispers" reflects the former stereotype in Europe of the Chinese language as being incomprehensible. Although it is sometimes considered offensive in the US, it remains the common British English name for the game and is not generally regarded as being offensive.

colimit ontology (i.e. alignment) as matching O_2 with O_1[9]; b) matching algorithms are 'not transitive', i.e., matching $\langle O_1, O_2\rangle$ and $\langle O_2, O_3\rangle$ and computing the colimit yields, in general, a different result than matching and aligning $\langle O_1, O_3\rangle$.

With these assumptions in place, given a repository $\mathfrak{R}$ with N ontologies, we start with $l = \frac{N\times(N-1)}{2}$ matching pairs.[10] Playing chinese whispers on $\mathfrak{R}$ with $k \leq N$ players now means to pick a connected subgraph of the hyperontology graph (to ensure that each ontology 'talks' to at least one other), which we call a **matching configuration** .

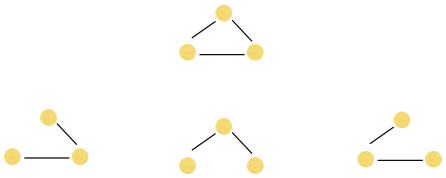

Fig. 3. The number of non-isomorphic matching configurations for $N = 3$

Given a fixed k, the largest possible matching configuration (measured in pairs of matched ontologies) corresponds to a clique with k nodes, i.e. a complete subgraph of the hyperontology graph with k nodes.

We have seen that in practise, not every pair of ontologies will match at all, i.e. a matcher will report no synonyms. Therefore, for fixed $k \leq N$ ontologies $O_1, \ldots, O_k$, the number of non-isomorphic matching configurations containing the O_i $(i = 1, \ldots, k)$ corresponds to the number of connected components of the hyperontology graph with these ontology nodes. The case of $N = 3$ (assuming a clique) is illustrated in Fig. 3. The alignment operation on a matching configuration, i.e. the computation of the colimit of that graph, we call a **connected alignment**.

4 Modularity in Hyperontologies

OK: What's up with ORATE?

• ORATE [17]

OK: Is the footnote at the right place?

In the following, we will make precise what we mean by a module and define the notion of conservativity. We start with some auxiliary notions. Let Σ be a signature containing concept names and roles.[11] et $\mathbf{Sen}(\Sigma)$ be the set of sentences formulated using the symbols in Σ in some language. The language depends on the language the ontologies under consideration are formulated in, e.g. $\mathcal{OWL}$ or some fragment thereof.

For defining deductive conservativity, we first introduce the notion of a Σ-theory of an ontology O:

$$Th_\Sigma(O) = \{\phi \in \mathbf{Sen}(\Sigma) \mid O \models \phi\}.$$

[9] Whether or not this holds for actual matching systems is an implementational artefact which we ignore; the assumption is certainly reasonable to make as both 'agreement' and 'synonymy' are symmetric.

[10] We assume that we do not match ontologies with themselves.

[11] We use the DL terminology *concept name* and *role* interchangeably with the $\mathcal{OWL}$ terminology *class* and *property*.

Definition 1 (Consequence-theoretic Conservativity). *Given two ontologies O_1, O_2 and a signature Σ, we say that O_2 is a* consequence-theoretic Σ-conservative extension *of O_1 if $Th_\Sigma(O_2) \subseteq Th_\Sigma(O_1)$.*

Notice that we have $Th_\Sigma(O_1) \subseteq Th_\Sigma(O_2)$ for any $O_1 \subseteq O_2$ that are formulated in a monotonic logic.

Definition 2 (Model-theoretic Conservativity). *Given two ontologies O_1, O_2 and a signature Σ, we say that O_2 is a* model-theoretic Σ-conservative extension *of O_1 if for all models $\mathcal{I}$ of O_1, there exists a model $\mathcal{J}$ of O_2 such that $\mathcal{I}{\restriction_\Sigma} = \mathcal{J}{\restriction_\Sigma}$.*

Note that $\mathcal{I}{\restriction_\Sigma} = \mathcal{J}{\restriction_\Sigma}$ states that both models are equivalent when each is restricted to the symbols in Σ.

The notion of model-theoretic conservativity is stronger than consequence-theoretic conservativity. To be precise, the former implies the latter, but not vice versa [13]. The two notions coincide if we define consequence-theoretic conservativity using Σ-theories that contain consequences $\phi \in \mathbf{Sen}(\Sigma)$ formulated in Second-Order Logic.

The computational complexity of deciding conservativity appears to be rather daunting even if the ontologies are formulated in weak logics. For instance, for ontologies formulated in the light-weight Description Logic $\mathcal{EL}$, deciding consequence-theoretic conservativity is ExpTime-complete, and model-theoretic conservativity is undecidable. The former problem also becomes undecidable when adding nominals to $\mathcal{ALCIQ}$, for which it is still 2-ExpTime-complete [15].

We continue with introducing a general notion of a module in the sense that a module of an ontology is not restricted to be a subset of the ontology. It is crucial, however, that the module does not say anything that is not already said by the ontology itself (i.e. the module is required to be a conservative extension of the ontology). We say that a module covers a signature of interest when it says the same about that signature as the entire ontology. The next definition makes these intuitions precise.

Definition 3 (Module Generator). *Let O be an ontology, and let $\Sigma \subseteq \mathsf{Sig}(O)$ be a signature. A function*

$$\Pi : \langle O, \Sigma \rangle \mapsto \mathbf{Sen}(\mathsf{Sig}(O))$$

mapping pairs $\langle O, \Sigma \rangle$ consisting of an ontology O together with a signature Σ to a set of sentences in $\mathsf{Sig}(O)$ is called a ***Σ-module generator*** *if for all O and Σ:*

$$\Pi(\langle O, \Sigma \rangle) \text{ is a consequence-theoretic } \mathsf{Sig}(O)\text{-conservative extension of } O.$$

For a Σ-module generator Π, the set $\Pi(\langle O, \Sigma \rangle)$ is called a ***Σ-module for*** *O. $\Pi(\langle O, \Sigma \rangle)$ is called* ***Σ-covering for*** *O if:*

$$O \text{ is a consequence-theoretic } \Sigma\text{-conservative extension of } \Pi(\langle O, \Sigma \rangle).$$

OK: Explain why we use different terminology (heterogeneity) as known in DL literature, i.e. safety and coverage.

We now illustrate the notion of a module with some examples.

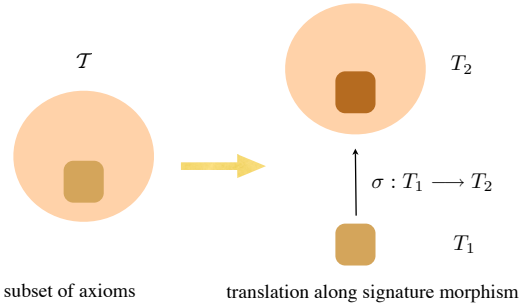

Fig. 4. Modules as subsets vs. modules as image under translation.

Example 2. (1) Obviously, the simplest possible module generator is the function Π_{id} mapping each pair $\langle O, \Sigma\rangle$ to O. We have that $\Pi_{\text{id}}(\langle O, \Sigma\rangle)$ is a Σ -module for O and that it is Σ-covering for O.

(2) Assume the consequences $O \models \phi, \phi \in \mathbf{Sen}(\Sigma)$ (a countable set) are well-ordered by ω. Define the following

$$\Pi_{\mathsf{ord}}(\langle O, \Sigma\rangle) = \{\phi_0, \ldots, \phi_n \mid n \text{ is minimal such that } \{\phi_i \mid i \leq n\} \text{ is conservative}\}$$

OK: check examples

By construction, Π_{ord} is a conservative module generator. Since O is finite, there are only finitely many (up to exponentially many in the size of O) well-orderings such that Π_{ord} is non-creative, and a continuum of creative ones. (3) Define $\mathsf{Con} := \{O' \mid O' \subseteq \mathbf{Sen}(\mathsf{Sig}(O))$ and for all $\phi \in \mathbf{Sen}(\Sigma) : O \models \phi \implies O' \models \phi\}$. Choose some $O' \in \mathsf{Con}$ and set $\Pi_{\mathsf{con}}(\langle O, \Sigma\rangle) = O'$. This construction always returns a non-creative and conservative module generator, but only sometimes a covering one.

OK: What is the right level of abstraction for our multiple alignment application: certainly we need to consider super-signatures as we merge ontologies.

(4) A variation of the last example, and one that we will use in detail below, is to use syntactic checks to establish whether we need to add an axiom to a Π-reduction. More specifically, we will employ locality-based modules [20,10] as their computation is readily supported by the $\mathcal{OWL}$-API. Call a function $\lambda_\Sigma : \mathbf{Sen}(\mathsf{Sig}(O)) \to 2 = \{0, 1\}$ Σ**-safe** if the module generator Π_{apx} defined via

$$\Pi_{\mathsf{apx}}(\langle O, \Sigma\rangle) = \{\phi \mid \phi \in O \text{ and } \lambda_\Sigma(\phi) = 1\}$$

is Σ-conservative.

OK: define and proof

The idea to use entrapping module generators is to massively reduce the size of a colimit ontology whilst preserving the semantics completely. We will give examples below.

Lemma 2 (Locality based modules). *Locality based modules are entrapping module operators.*

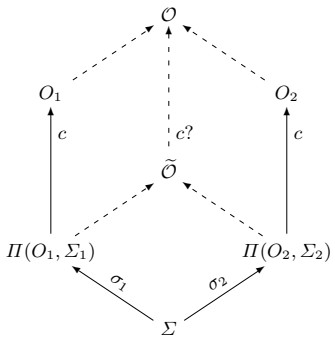

Fig. 5. Propagation of modular structure through one matching

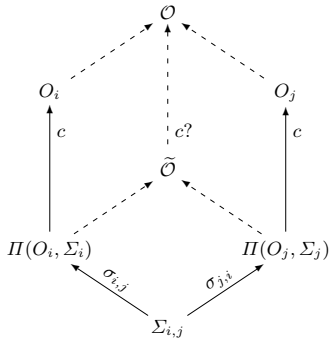

Fig. 6. Propagation of modular structure through matching network

ule propaga-
introduce
liscuss

Proof. Show that locality-based modules produce entrapping modules.

Theorem 1 (Combination of two modules). *Assume a semi-exact institution. Consider Fig. 5, and assume that $\widetilde{\mathcal{O}}$ and $\mathcal{O}$ are obtained by pushouts (with base Σ). Then $\mathcal{O}$ is a model-theoretic conservative extension of $\widetilde{\mathcal{O}}$. In particular, $\mathcal{O}$ is satisfiable iff $\widetilde{\mathcal{O}}$ is.*

Proof. Let $\widetilde{M}$ be a model of $\widetilde{\mathcal{O}}$. For $i = 1, 2$, let M_i be an O_i-expansion of the $\Pi(O_1, \Sigma_i)$-reduct of $\widetilde{M}$ (which exists by conservativity of O_i over $\Pi(O_1, \Sigma_i)$). M_1 and M_2 obviously agree on Σ, hence they have an amalgamation M which is an $\mathcal{O}$-model. Now the $\Pi(O_1, \Sigma_i)$-reducts of M agree with those of $\widetilde{M}$, hence by uniqueness of amalgamation, the $\widetilde{\mathcal{O}}$-reduct of M is $\widetilde{M}$. ⊣

OK: TM: Should we work for arbitrary institutions, or in a fixed one? All usual DLs are semi-exact and have an initial signature, see our ECAI paper.

Theorem 2 (Combination of multiple modules). *Assume a semi-exact institution with an initial signature. Consider a family of ontologies $(O_i)_{i \in I}$ indexed by a finite non-empty set I and a loop-free connected symmetric graph $G \subseteq I \times I$, such that for $(i, j) \in G$,*

OK: TM: We can use unions instead of pushouts, if all arrows are inclusions and Σ is the intersection of O_1 and O_2. However, I doubt that this is the case, since this would enforce a "same name - same concept" principle, contradicting the remarks about synonymy

O_i and O_j are interfaced by

$$O_i \xleftarrow{\theta_{i,j}} \Sigma_{i,j} \xrightarrow{\theta_{j,i}} O_j$$

Define

$$\Sigma_i := \bigcup_{j \in I \setminus \{i\}} \theta_{i,j}(\Sigma_{i,j})$$

and $\sigma_i \colon \Sigma_i \to \Pi(O_i, \Sigma_i)$ the module in O_i for Σ_i. Let $\sigma_{i,j} \colon \Sigma_{i,j} \to \Pi(O_i, \Sigma_i)$ be the restriction of $\theta_{i,j}$, namely $\theta_{i,j} \colon \Sigma_{i,j} \to \Sigma_i$, composed with σ_i, see Fig. 6. Assume that $\widetilde{\mathcal{O}}$ (resp. $\mathcal{O}$) is obtained by the colimit of the diagram of all $\sigma_{i,j}$ (resp. all $\sigma_{i,j}$ composed with the inclusion of $\Pi(O_i, \Sigma_i)$ into O_i). Then $\mathcal{O}$ is a model-theoretic conservative extension of $\widetilde{\mathcal{O}}$. In particular, $\mathcal{O}$ is satisfiable iff $\widetilde{\mathcal{O}}$ is.

Proof. By Prop. 4.4.15 of [19], in any semi-exact institution with an initial signature, all finite non-empty connected diagrams enjoy the amalgamation property. With this, the proof is a straightforward generalisation of the proof of Thm. 1. Note that connectedness of G ensures connectedness of the diagrams for obtaining $\widetilde{\mathcal{O}}$ and $\mathcal{O}$. ⊣

Note that Thm. 1 does not hold for consequence-theoretic conservativity. Consider the following example, adpated from [14]. In $\mathcal{ALCO}$, let O_1 be

$$\begin{aligned} &\mathsf{IntroTCS} && \sqsubseteq \exists \mathsf{has\_subject.AutomataTheory} \\ &\mathsf{IntroTCS} && \sqsubseteq \exists \mathsf{has\_subject.ComplexityTheory} \\ &\mathsf{AutomataTheory} \sqcap \mathsf{ComplexityTheory} && \sqsubseteq \bot \end{aligned}$$

and O_2 be

$$\begin{aligned} \mathsf{IntroTCS} &\sqsubseteq \forall \mathsf{has\_subject.}\{\mathsf{moore\_automata}\} \\ \mathsf{IntroTCS} &\sqsubseteq \exists \mathsf{has\_subject.}\{\mathsf{moore\_automata}\} \end{aligned}$$

Let Σ be $\{\mathsf{IntroTCS}, \mathsf{has\_subject}\}$. Then assuming a consequence-theoretically conservative (minimal) module generator, $\Pi(O_1, \Sigma) = \Pi(O_2, \Sigma) = \widetilde{\mathcal{O}}$ is

$$\mathsf{IntroTCS} \sqsubseteq \exists \mathsf{has\_subject.} \top$$

But this is consistent, while $O_1 \cup O_2$ is not.

IN: Check whether FALCON is symmetric or not?

IN: Abbreviations for systems....

5 An Interoperability Workflow and Prototypical Implementation

5.1 The Component Tools

Matching. FALCON [8]
Reasoning. Pellet [21], Racer [6], FaCT++ [22].
Modularity. OWL-API [18]
Structuring and Iterative Matching. Hets [7]

5.2 Workflow Description

Our workflow of multiple ontology alignment consists of two phases: 1) the preprocessing of the whole repository to a complete list of pairwise **matching records** and 2) the alignment of a (user selected) connected **hyperontology graph**. The meaning of the two technical terms "matching record" and "hyperontology graph" will become clear in the following. Fig. 5.2 shows the whole workflow in pseudo code. We are going to explain it now line by line. Procedure `preprocess_repository` takes each pair of ontologies `(o1,o2)` and applies the procedure `match_ontologies` to it, i.e., it matches pairwise all ontologies from the repository. The output of `match_ontologies` is a `matching_record` more than just the mapping between two ontologies: the matching system FALCON computes the `mapping` between the two ontologies `o1` and /verb+o2+. Depending on predefined configuration the `mapping` is revised to `mapping'` (e.g. by dismissing mappings whose confidence is below a given threshold). From the revised `mapping'` the concept `cs1` (`cs2`) belonging to ontology `o1` (`o2`) are extracted. Based on the concepts `cs1` (`cs2`) the module `m1` (`m2`) is extracted from ontology `o1` (`o2`). The `mapping'`, the concepts `cs1` and `cs2`, the corresponding modules `m1` and `m2` are packed into a record `matching_record` equipped with an entry `"o1_to_o2"` to reconstruct later on which ontologies were matched in this record. A `matching_record` can be viewed as a link between two ontologies. By applying `match_ontologies` pairwise on all ontologies in the repository `preprocess_repository` we link all ontologies with each other. We call this network whose nodes are ontologies and whose edges are the matching records *hyperontology graph*. Although all ontologies are matched pairwise the graph is not complete, i.e., some pairs of ontologies (in practise even the majority) are not linked, namely when the matching system cannot find any mappings.

Once the hyperontology graph of the ontology repository is computed we can align any combination of modules that is stored in the hyperontology graph during the preprocessing phase. The `align_modules_from_graph` procedure specifies how this is done: its input is a subgraph of the hyperontology graph. This subgraph determines which modules (its nodes) and which matching records (its edges) should be taken into account in the alignment.

```
preprocess_repository = {
  foreach (o1,o2) in repository
   match_ontologies o1 o2
  end
}

match_ontologies o1 o2 = {
  mapping = falcon o1 o2
  mapping' = revise mapping
  cs1 = extract_concepts_from_o1 mapping'
  cs2 = extract_concepts_from_o2 mapping'
  m1 = extract_module cs1 o1 % pellet modularity --signature o1_concepts o1
  m2 = extract_module cs2 o2
  matching_record = ("o1_to_o2",cs1,cs2,m1,m2,mapping')
  if (not_empty mapping') store_in_hyperontology_graph matching_record
```

```
}

align_modules_from_graph hyperontology_subgraph = {
  modules = get_ontologies hyperontology_subgraph
  interfaces = {compute_interface edge | edge in hyperontology_subgraph}
  views = {compute_views edge | edge in hyperontology_subgraph}
  spec = write_V_alignment_spec ontologies interfaces views
  V_alignment = compute_V_alignment spec % hets
}
```

5.3 Selecting Reusable Knowledge from the Hyperontology Graph

IN: tolle Namen ;-)

IN: schon definiert?

In this section we describe how an ontology developer Otto can make reuse of appropriate modules from an repository OOR in order to develop a new ontology Onto. Let us call the tool for this support **ontology composer** Before any interaction between Otto and the ontology composertakes place, the whole OOR is preprocessed by the matching system: each ontology from the OOR is matched against each other thus building the hyperontology graphFor our current experiments we use FALCON as matching system, because it can be used in a batch mode and thus it can be easily plugged into the whole process. Once the hyperontology graph is calculated Otto starts working on the **hyperontology graph editor** to identify those parts of the graph that are relevant for Onto. More specifically this means to identify those concept names in the graph that should occur in Onto. Through the synonymy links between concept nodes these input concept names will be connected to various other concept names not explicitly selected by Otto. Since matching systems are never completely reliable Otto needs to repair the focused fragment of the hyperontology graph; i.e., some of the automatically generated synonym links are wrong and thus should be deleted manually. Other pairs of concept names are not recognised by the matching system, hence synonym links can be also added to the graph in the hyperontology graph editor. A detailed design discussion of the graphical user interface of such a hyperontology graph editor is given in [16]. When Otto has completed the adjustment on the hyperontology graph the provided fragment of the graph is used to induce those modules from ontologies in the repository that will be taken to create Onto: for each ontology node all directly connected concept nodes are the concepts that are used as input for the module extractor. The extracted modules and the synsetsf the hyperontology graph form the input for the final the V-alignment: for each synset an arbitrary concept name is selected (or alternatively determined by the user) and from that singled out concept name to each other concept name of this synset a signature morphism is generated to each ontology where that other concept name belongs to—Fig. 7 shows an example.

IN: defined?

IN: Statistiken zum matching in OASIS hier hin?

6 Merging

After the matching of two or more ontologies, we obtain an alignment that contains correspondences between the signatures of the input ontologies. The alignment can now

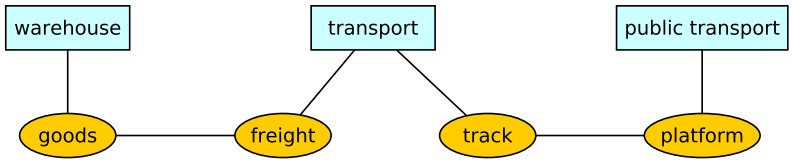

Fig. 7. Fragment of a hyperontology graph inducing an interface for a V-alignment. Blue rectangles = ontology nodes, yellow ellipses = concept nodes. Edge between concept nodes = synonym.

```
spec Interface =
  Class: goods
  Class: track
end

view v1 : Interface to transportation =
    goods |-> freight
end

view v2 : Interface to public_transport =
    track |-> platform
end
```

Fig. 8. Induced interface and signature morphisms (=view in CASL notation) from the hyperontology graph in Fig. 7

be used for data translation or merging. In this section, we are interested in the latter. and, in particular, we are interested in how

In [1] the authors introduce and investigate applications of ontology re-use under the notion of safety of an ontology.

Definition 4 (Safety wrt. an ontology [1]). *Given two ontologies O_1, O_2, we say that O_1 can safely be imported into O_2 if $O_1 \cup O_2$ is a deductive Σ-conservative extension of O_1 with $\Sigma = \mathsf{Sig}(O_1)$.*

Intuitively, O_1 can safely be imported into O_2 states that the meaning of the terms defined in O_1 does not change when O_1 is put into the context of O_2.

Definition 5 (Safety wrt. a signature [1]). *Given an ontology O_2 and a signature Σ, we say that O_2* is safe for *Σ if O_1 can safely be imported into O_2, for all ontologies O_1 with $\mathsf{Sig}(O_1) \cap \mathsf{Sig}(O_2) \subseteq \Sigma$.*

That is, any changes in the external ontology O_1 wont conflict with the locally developed ontology O_2, because O_2 does not "say anything" about the terms defined in O_1.

In other words, O_2 says nothing about the terms in Σ; formally, O_2 is a model Σ-conservative extension of $\varnothing$.

DW: Relationship between Matching and Safety: after matching 'imported ontology' is safe for union...

Locality.. syntactic condition that is sufficient (but not necessary) to ensure safety.

In [9] a semi-automatic procedure is presented for the integration of ontologies that involves revision of mappings. This approach is implemented as the Protégé 4 plugin ContentMap. At first a selected ontology matcher is used to compute mappings between signature symbols of the two ontologies that are to be integrated. The mappings are explicitly represented as axioms in an OWL 2 ontology. The result of the integration is taken to be the union of the original ontologies together with axioms for the mappings. The integration said to be successful if there are no unintended logical consequences. It is then up to the user to identify the unintended consequences. This decision is guided by justifications (explanations for entailment) that can automatically be computed, e.g., by ContentMap and the confidence values created by the matcher for the mapping axioms. To eliminate the unintended consequences, a repair plan is created describing which axioms from the original ontologies or the mapping should be removed. It should be noted that such a plan does not always exist and that a desired integration may require the iteration of these steps.

– visualisation

7 Outlook

We could only scratch the surface of the area of problems related to matching in networks of ontologies. We have laid out the engineering infrastructure...

Semantic matching; similarity $\mathcal{E}$-connection as internalising the confidence values in matchings?

full statistical analysis of major ontologies and repositories is future work Chinese whispers: results for related ontologies, results for randomised selections... magic inconsistency numbers.

References

1. Cuenca Grau, B., Horrocks, I., Kazakov, Y., and Sattler, U. Modular reuse of ontologies: Theory and practice. *J. of Artificial Intelligence Research (JAIR) 31* (2008), 273–318.
2. David, J., and Euzenat, J. On fixing semantic alignment evaluation measures. In *OM* (2008), P. Shvaiko, J. Euzenat, F. Giunchiglia, and H. Stuckenschmidt, Eds., vol. 431 of *CEUR Workshop Proceedings*, CEUR-WS.org.
3. Euzenat, J. Semantic precision and recall for ontology alignment evaluation. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, January 6-12, 2007* (2007), M. M. Veloso, Ed., pp. 348–353.
4. Goguen, J. A., and Burstall, R. M. Introducing institutions. In *Proc. Logics of Programming Workshop* (1984), E. Clarke and D. Kozen, Eds., vol. 164 of *LNCS*, Springer, pp. 221–256.
5. Goguen, J. A., and Burstall, R. M. Institutions: Abstract Model Theory for Specification and Programming. *Journal of the ACM 39* (1992), 95–146.
6. Haarslev, V., and MŽller, R. RACER system description. In *Proceedings of IJCAR'2001* (2001), Springer-Verlag, Berlin, pp. 701–705.
7. Hets - the Heterogeneous Tool Set. `www.informatik.uni-bremen.de/cofi/hets`.

8. Hu, W., and Qu, Y. Falcon-ao: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web 6*, 3 (2008), 237 – 239. World Wide Web Conference 2007, Semantic Web Track.
9. Jimenez Ruiz, E., Cuenca Grau, B., Horrocks, I., and Berlanga, R. Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In *Proc. of the 6th European Semantic Web Conference (ESWC 2009)* (2009), LNCS, Springer.
10. Kutz, O., Hois, J., Bao, J., and Grau, B. C., Eds. *Modular Ontologies—Proceedings of the Fourth International Workshop (WoMO 2010)*, vol. 210 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Toronto, Canada, 2010.
11. Kutz, O., Mossakowski, T., and Codescu, M. Shapes of Alignments: Construction, Combination, and Computation. In *Proc of the 1st Workshop on Ontologies: Reasoning and Modularity (WORM-08)* (ESWC, Tenerife, Spain, 2008), U. Sattler and A. Tamilin, Eds., CEUR-WS, Vol-348.
12. Kutz, O., Mossakowski, T., and Lücke, D. Carnap, Goguen, and the Hyperontologies—Logical Pluralism and Heterogeneous Structuring in Ontology Design. *Submitted to a Journal* (2010). `http://www.informatik.uni-bremen.de/~okutz/hyperontologies.pdf`.
13. Lutz, C., Walther, D., and Wolter, F. Conservative Extensions in Expressive Description Logics. In *Proceedings of IJCAI 2007: the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007* (2007), M. M. Veloso, Ed., AAAI Press, pp. 453–458.
14. Lutz, C., Walther, D., and Wolter, F. Conservative extensions in expressive description logics. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)* (2007), M. Veloso, Ed., AAAI Press, pp. 453–458.
15. Lutz, C., and Wolter, F. Conservative Extensions in the Lightweight Description Logic $\mathcal{EL}$. In *Proceedings of the 21st Conference on Automated Deduction (CADE-21)* (2007), F. Pfenning, Ed., vol. 4603 of *Lecture Notes in Computer Science*, Springer, pp. 84–99.
16. Normann, I., and Kutz, O. Ontology Reuse and Exploration via Interactive Graph Manipulation. *Submitted* (2010). `http://www.informatik.uni-bremen.de/~okutz/visualisation.pdf`.
17. ORATE – Ontology Repository Assistive TEchnologies. `http://ontologies.informatik.bremen.de`.
18. The OWL API. `http://owlapi.sourceforge.net`.
19. Sannella, D., and Tarlecki, A. *Foundations of Algebraic Specifications and Formal Program Development*. Springer Verlag. to appear.
20. Sattler, U., Schneider, T., and Zakharyaschev, M. Which kind of module should I extract? In *Proc. of DL-09* (2009), vol. 477, CEUR-WS.
21. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., and Katz, Y. Pellet: A practical OWL DL reasoner. *Journal of Web Semantics 5*, 2 (2007), 51–53.
22. Tsarkov, D., and Horrocks, I. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)* (2006), vol. 4130 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 292–297.
23. Zimmermann, A., Krötzsch, M., Euzenat, J., and Hitzler, P. Formalizing Ontology Alignment and its Operations with Category Theory. In *Proc. of FOIS-06* (2006), pp. 277–288.

TM: Space for Till to write up fancy diagrams with all bells and whistles

Appendix