IF-Map: An Ontology-Mapping Method based on Information-Flow Theory

Yannis Kalfoglou[1] and Marco Schorlemmer[2,3] *

Advanced Knowledge Technologies

[1] Department of Electronics and Computer Science
University of Southampton

[2] Centre for Intelligent Systems and their Applications
School of Informatics
The University of Edinburgh

[3] Escola Universitària de Tecnologies d'Informació i Comunicació
Universitat Internacional de Catalunya

Abstract. In order to tackle the need of sharing knowledge within and across organisational boundaries, the last decade has seen researchers both in academia and industry advocating for the use of ontologies as a means for providing a shared understanding of common domains. But with the generalised use of large distributed environments such as the World Wide Web came the proliferation of many different ontologies, even for the same or similar domain, hence setting forth a new need of sharing—that of sharing ontologies. In addition, if visions such as the Semantic Web are ever going to become a reality, it will be necessary to provide as much automated support as possible to the task of mapping different ontologies. Although many efforts in ontology mapping have already been carried out, we have noticed that few of them are based on strong theoretical grounds and on principled methodologies. Furthermore, many of them are based only on syntactical criteria. In this paper we present a theory and method for automated ontology mapping based on channel theory, a mathematical theory of semantic information flow. We successfully applied our method to a large-scale scenario involving the mapping of several different ontologies of computer-science departments from various UK universities.

1 Introduction

The wide-spread use of ontologies by diverse communities and in a variety of applications is a commonality in today's knowledge-sharing efforts. They are the backbone for semantically-rich information sharing, a prerequisite for knowledge sharing. As systems become more distributed and disparate within and across organisational boundaries, there is a need to preserve the meaning of concepts

* Last names of authors are in alphabetical order.

used in everyday transactions of information sharing. The emergence of the Semantic Web [5] has made these transactions, arguably, easier to implement and deploy on a large scale in a distributed environment like the Internet. However, at the same time poses some interesting challenges. For instance, we observe that the demand for knowledge sharing has outstripped the current supply. And even when knowledge sharing is feasible, this is only within the boundaries of a specific system, when certain assumptions hold, and within a specific domain. The reason for this shortcoming is, probably, the very environment and technologies that generated a high demand for sharing: The more ontologies are being deployed on the Semantic Web, the more the demand to share them for the benefits of knowledge sharing and semantic interoperability. The sharing of ontology though, is not a solved problem. It has been acknowledged and researched by the knowledge engineering community for years.

In fact, this problem is related to that of integration of heterogeneous databases conducted in the 1980s and early 1990s [4, 39], as far as database schemas can be considered a particular kind of lightweight ontologies. These early efforts in information sharing run into the problem of semantic heterogeneity, which required the identification and representation of all semantics useful in performing schema translation and schema integration. Current efforts on ontology sharing build upon the previous experience, but in the light of more broader a concept of ontology, and of more expressive logical formalisms and theories that better capture the semantics and information flow.

One of the aspects in ontology sharing is to perform some sort of mapping between ontology constructs. That is, given two ontologies, one should be able to map concepts found in one ontology onto the ones found in the other. Further, some research suggest that we should also be able to combine ontologies where the product of this combination will be, at the very least, the intersection of the two given ontologies. These are the dominant approaches that have been studied and applied in a variety of systems (see, for example, [35]).

There are, however, some drawbacks that prevent engineers from benefitting from such systems. Firstly, the assumptions made in devising ontology mappings and in combining ontologies are not always exposed to the community and no technical details are disclosed. Secondly, the systems that perform ontology mapping are often either embedded in an integrated environment for ontology editing or are attached to a specific formalism. Thirdly, in most cases mapping and merging are based on heuristics that mostly use syntactic clues to determine correspondence or equivalence between ontology concepts, but rarely use the meaning of those concepts, i.e., their semantics. Fourthly, most, if not all approaches, do not treat ontological axioms or rules often found in formal ontologies. Finally, ontology mapping as a term has a different meaning in different works merely due to the lack of a formal account of what ontology mapping is. There is an observed lack of theory behind most of the works in this area.

Motivated by these drawbacks we started to work on a method and a theory for ontology mapping and merging. We were determined to tackle these drawbacks so our approach is based on the observation reported in [38] of an essential

duality in knowledge sharing, namely that sharing occurs both at the token and at the type level. We draw on sound theoretical ground, but at the same time we are providing a systematic approach for ontology mapping in mechanised and methodological steps.

In particular, we propose an *Information-Flow-based* method for ontology mapping (hereafter, IF-Map). We are mostly interested in mapping ontologies, but we could extend the approach to merge them, too. IF-Map draws from the works of Schorlemmer on aligning ontologies [38] and on the heuristics defined by Kalfoglou (in [20], pp.95–97), to analyse prospective mappings between ontologies. On the theoretical side, our method is based on the mathematical theory of information flow proposed by Barwise and Seligman [3] and on Kent's Information Flow Framework for the IEEE standardisation activity of an upper-level ontology [23], and also on his proposed methodology for merging ontologies [22, 24]. The methodological part of IF-Map has also been influenced by the work of Stumme and Maedche on the FCA-Merge method [41].

We outline a scenario and describe the architecture we have built to perform ontology mapping in Section 2. We briefly provide mathematical preliminaries on information flow and channel theory in Section 3, before we proceed to describe our ontology-mapping method in Section 4, together with an example case of its use. In Section 5 we report on an application of our method and elaborate on scalability issues. We discuss related work in Section 6 and summarise the paper in Section 7.

2 A Scenario and Architecture for Ontology Mapping

We should clarify some terminological issues before we continue with our architecture for ontology mapping. Some researchers view the mapping process as an integral part of alignment or merging. For example Noy and Musen view merging as the creation of "a single coherent ontology that includes the information from all the sources" and alignment as a process in which "the sources must be made consistent and coherent with one another but kept separately" [34]. Clearly, mapping is an essential aspect of alignment and could also be used to initiate merging. In this paper, we adopt a working view of ontology mapping. We do not intend to alter the original ontologies but rather look for possible mappings between them. To that extent we are compatible with Noy and Musen's definition of alignment, however, we view alignment as a process that involves merging but without altering the original ontologies. This would be done by constructing a *virtual* ontology consisting of the mapped constructs of the two input ontologies. This view is more akin to those researchers working on the IEEE SUO initiative—the standardisation of an *upper level ontology*—where a meta-level framework for relating ontologies based on the Barwise-Seligman theory of information flow is currently being developed.[1] Although on our scenario we have not yet implemented this aspect, we could extend the existing IF-Map

[1] We point the interested reader to `http://suo.ieee.org` for more information on this initiative.

method to perform merging. In the remaining of the paper though, we will focus on ontology mapping only.

2.1 Ontology Mapping Scenario

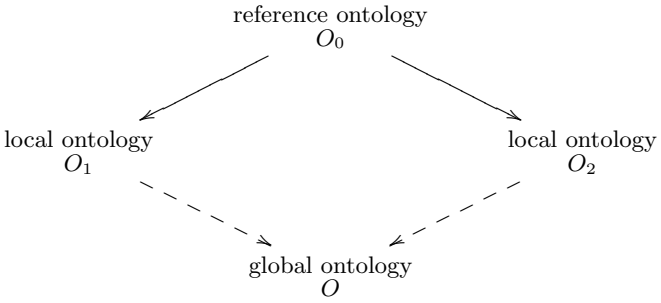


Fig. 1. The scenario for ontology mapping.

In Figure 1 we illustrate our approach to ontology mapping. In particular, the focus is on the use of information-flow theory as the underpinning mathematical foundation for establishing mappings between two ontologies. We shall formalise these mappings in terms of *local logics* and *logic infomorphisms*, which we introduce in Section 3.

We assume that, when two communities desire to share knowledge but each one has encoded knowledge according to its own *local ontology*—O_1 and O_2 respectively—they have previously agreed upon a common understanding, a *reference ontology* O_0, in order to favour the sharing of knowledge. Typically each community will have its local ontology populated with its own instances, while the reference ontology will not have instances.

But, although having agreed upon a reference ontology, each community might prefer to communicate via its own local ontology, provided these ontologies O_1 and O_2 (or at least a significant fragment of them) conform to the reference ontology O_0. In this paper we provide a methodology to establish this conformance by looking for, and further generating, ontology mappings $O_0 \rightarrow O_1$ and $O_0 \rightarrow O_2$ from the reference ontology to local ontologies, denoted as solid arrows in Figure 1. The methodology is based upon formalising ontologies as *local logics* and ontology mappings as *logic infomorphisms*, as we shall see in Section 3.5.

The link $O_1 \leftarrow O_0 \rightarrow O_2$ between local ontologies via the reference ontology, together with the generated ontology mappings from the latter to the former two, provides an *alignment structure* of ontologies that determine uniquely a *global ontology* O—an ontology that could be either constructed from the alignment structure or else generated 'on-the-fly' for the purpose of knowledge sharing. The

dashed arrows denote the inclusion of local ontologies O_1 and O_2 into the global ontology O. This inclusions are also ontology mappings as formalised by logic infomorphisms, and hence our methodology IF-Map could be extended in future to include the merging of ontologies according to an alignment structure.

Figure 1 clearly resembles Kent's proposed two-step process for conceptual knowledge organisation [22, 24]. In fact, the existence of a unique global ontology O given an alignment $O_1 \leftarrow O_0 \rightarrow O_2$ consisting of local ontologies and a reference ontology is the consequence of formalising ontologies and ontology mappings as local logics and logic infomorphisms.

2.2 Ontology Mapping Architecture

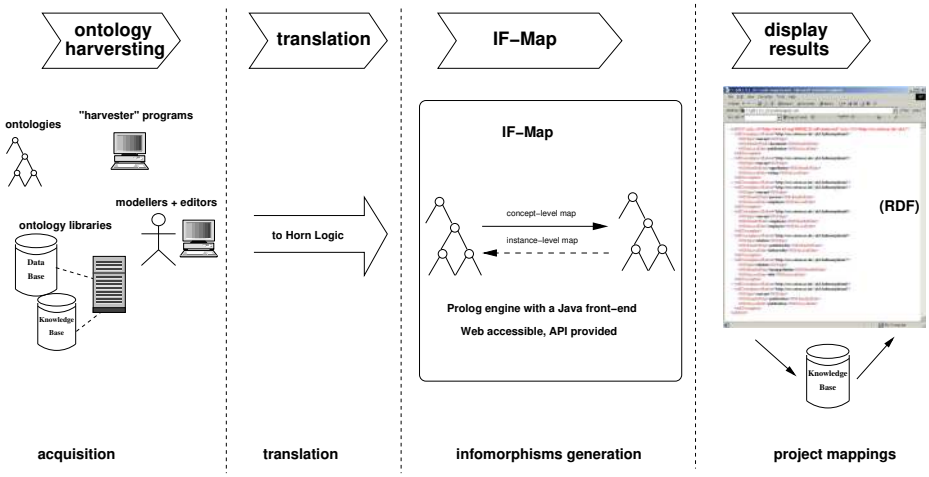


Fig. 2. The IF-Map architecture.

In Figure 2 we illustrate the process of IF-Map. We have built a step-wise process that consists of four major steps: (a) Ontology harvesting, (b) translation, (c) infomorphism generation, and (d) display of results. In the ontology harvesting step we perform our acquisition. We acquire ontologies by applying a variety of methods: Downloading existing ontologies from ontology libraries (for example, from the Ontolingua [12] or WebOnto [10] servers); editing them in ontology editors (for example, in Protégé [16]); or extracting them from the Web. The latter is ongoing research in the AKT project (`http://www.aktors.org`), where we are writing scripting programs to crawl the Web and harvest RDF-encoded resources to semi-automatically construct and populate ontologies. We will not expand on this topic here as it is peripheral to our theme of ontology mapping. As a result of our versatile ontology acquisition step, we have to

deal with a variety of ontology representation languages ranging from KIF [15] and Ontolingua to OCML [31], RDF [27], Prolog, and native Protégé knowledge bases.

This introduces the second step in our process, that of translation. Since we have declaratively specified the IF-Map method in Horn logic and execute it with a Prolog interpreter, we partially translate the above formats to Prolog clauses. Our translator programs are either written in-house, or whenever available, we use public translators. For example, there are public RDF-to-Prolog translators[2] as well as Ontolingua-to-Prolog translators. In most of these cases though, we found it practical to write our own ones. We did that to have a partial translation, customised for the purposes of ontology mapping. Furthermore, as it has been reported in a large-scale experiment with publicly available translators [7], the Prolog code produced is not elegant or even executable. Our own translators are customised to translate—from KIF, Ontolingua, and Protégé knowledge bases into Prolog clauses—those constructs that are needed for IF-Map: Class taxonomy, relations and representative instances for classes. Thus, we deliberately neglect constructs such as documentation slots, separation of own-slots and template-slots and other object-oriented modelling primitives used in Ontology languages (such as KIF or Ontolingua[3]) as they are not useful for IF-Map and their absence from the translated Prolog code does not invalidate their meaning. For Protégé knowledge bases we used the built-in Java API to obtain the constructs we wanted, and for RDF we used publicly available RDF to Prolog translators. The issue of a full-blown translation from one formalism to another is a difficult problem, and Corrêa da Silva et al. [7] offer an account on the effort involved.

The next step in our process is the main mapping mechanism: The IF-Map method, which we describe in Section 4. We have written a Java front-end to the Prolog-written IF-Map program so that we can access it from the Web, and we are currently in the process of writing a Java API to enable external calls to it from other systems. This step will find logic infomorphisms, if any, between the two ontologies under examination, and in the last step of our process we display them in RDF format. This step involves translating back from Prolog clauses to RDF triples by means of an intermediary Java layer, where RDF is being produced using the Jena RDF API [28]. Finally, we store the results in a knowledge base for future reference and maintenance.

Before proceeding to an example case of the deployment this architecture we shall introduce the theoretical background of IF-Map. Therefore, in the next section we expand on logic infomorphisms and other channel-theoretic notions, and give a formal account of ontology mapping.

[2] Like the one from Wielemaker downloadable from `http://www.swi-prolog.org/packages/rdf2pl.html`

[3] We briefly describe the principles we used to partially translate from Ontolingua to Prolog in [20], pp.105–107.

3 Preliminaries

In order to give a formal characterisation of ontology mapping we start from the following basic assumption: If two communities desire to share information but their ontologies differ, we need to align and map their ontologies for the information to flow between them. Consequently, it would be desirable to base a theory of ontology mapping upon a mathematical theory that is capable of describing under which circumstances information flow occurs.

Although there is no such a theory yet, the most promising effort towards such a theory was initiated by Barwise and Perry with situation semantics [2], which was then been further developed by Devlin into a theory of information [8]. Barwise and Seligman's channel theory is currently the latest stage of this endeavour [3], and is probably the best place to start establishing a foundation for a theory of ontology mapping.

3.1 Information Flow

Barwise and Seligman propose a mathematical model that aims at establishing the laws that govern the flow of information. It is a general model that attempts to describe the information flow in any kind of distributed system, ranging form actual physical systems like a flashlight connecting a bulb to a switch and a battery, to abstract systems such as a mathematical proof connecting premises and hypothesis with inference steps and conclusions.

It is based on the understanding that information flow results from regularities in a distributed system, and that it is by virtue of regularities among the connections that information of some components of a system carries information of other components. The more regularities the system has, the more information flows; the more random the system is constituted the less information will be able to flow among its components.

As a notion of a component carrying information about another component, Barwise and Seligman follow Dretske's characterisation [11], in which assertions of the sort

> *The rifle shot carried the information that the king was dead to the whole city.*

or

> *The greyness of the sky carries the information that a storm is approaching.*

are abstractly characterised as following a common pattern

> *a's being F carries/bears/conveys the information that b is G.*

In fact, it is a particular rifle shot that will inform us of the king's death, and hence it is a particular event *token e* of the *type* 'rifle shot' that will carry this information. A rifle shot 50 years later will definitely not carry the information

that the present king is dead as will the present rifle shot not indicate that the king of the neighbouring city is dead. It is the fact that the event token of the rifle shot is connected spatio-temporarily with the event of the king's death that we know that the information flows. Channel theory formalises this crucial aspect of information flow, namely that it is the tokens and its connections that carry information, so that information flow involves both types and tokens.

3.2 Aligning and Merging Ontologies

With respect to the problem that concerns us here—that of mapping ontologies—the same abstract pattern arises. Two communities with different ontologies will be able to share information when they are capable of establishing connections among their tokens in order to infer the relationship among their types. Let us develop an example taken from [40], which shows the issues one has to take into account when attempting to align the English concepts river and stream with the French concepts fleuve and rivière. According to Sowa,

> In English, size is the feature that distinguishes river from stream; in French, a fleuve is a river that flows into the sea, and a rivière is either a river or a stream that runs into another river. [40]

This explains how the concepts need to be merged. Notice that the above quote requires an agreed understanding on how to distinguish between big and small rivers, and between rivers that run into a sea or into other rivers, yielding four types of instances of 'water-flowing entities': big-into-sea, big-into-river, small-into-sea, and small-into-river.

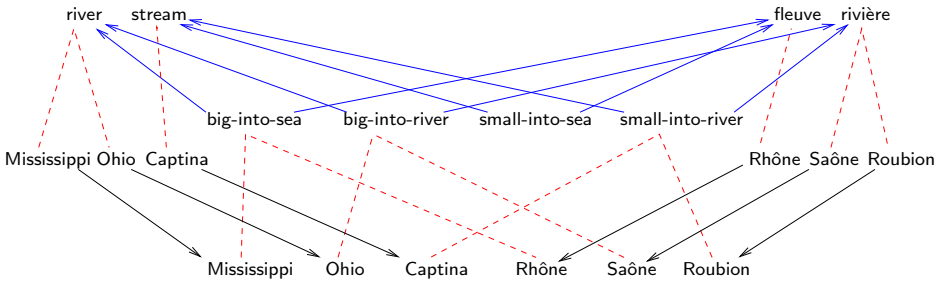


Fig. 3. Alignment by means of a pair of maps

Figure 3 shows how both, English and French speakers, base their concepts upon this agreed understanding, although English and French speakers don't distinguish between some types of instances. For example, English speakers call both, big-into-sea and big-into-river, a river, while French speakers don't distinguish between big-into-river and small-into-river, and call both types a rivière.

The agreed understanding is materialised by two maps that form the alignment. It requires the classification of particular instances of river, stream, fleuve, and rivière according with the agreed understanding, since it is this agreed way of classification which will determine how the concepts river, stream, fleuve, and rivière are going to be related to each other.

The ultimate goal is to determine the connections that link particular instances of type river or stream with particular instances of type fleuve or rivière, in a way that they respect the agreed understanding. This is done by connecting only those instances that conform to the same type according to the agreed understanding, as illustrated in Figure 4.

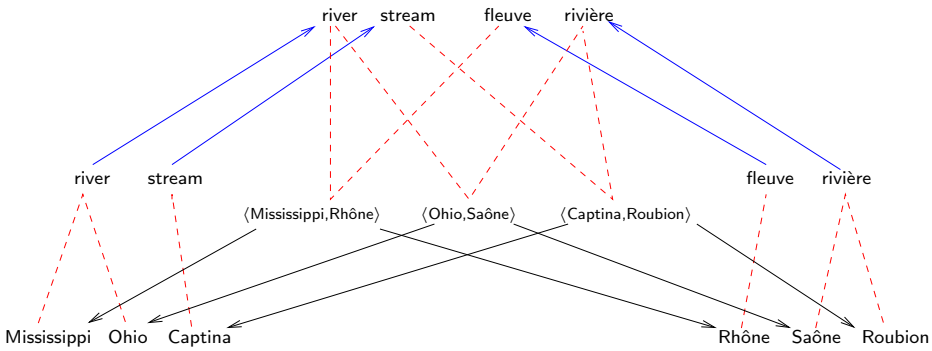


Fig. 4. Aligning ontologies through a pair of maps

The resulting classification of connections ⟨Mississippi, Rhône⟩, ⟨Ohio, Saône⟩, and ⟨Captina, Roubion⟩ into the four concepts river, stream, fleuve, and rivière, determines a theory of how these concepts are related (e.g., that a fleuve is also a river, or that a stream is also a rivière, but not vice versa). Figure 4 shows what in channel theory is known as a an *information channel* [3]. It captures, by means of two pairs of contra-variant functions, an existing duality between concepts and instances: Each pair consists of a map of concepts on the so called *type level* and map of instances on the so called *token level*, and pointing in the opposite direction. From a channel-theoretic perspective, Figure 4 actually illustrates us that sharing knowledge involves a flow of information that crucially depends on how the instances of different agents are connected together. The following table shows the *classification relation*, i.e., the connections as classified according to the concept types involved in the example:

	river	stream	fleuve	rivière
⟨Mississippi, Rhône⟩	1	0	1	0
⟨Ohio, Saône⟩	1	0	0	1
⟨Captina, Roubion⟩	0	1	0	1

The merged set of concepts {river, stream, fleuve, rivière} has an additional structure that we can deduce from the way the connections of instances are classified with respect to these concepts. Through techniques from formal concepts analysis [14], for instance, we can make such structure explicit in the form of a *concept lattice*, as shown in Figure 5. The concept hierarchy represented in this lattice depends on the choice of instances and its classification with respect to the agreed understanding. The fact that no instances were classified as of type small-into-sea was crucial in this example. Notice, also, that the resulting lattice has a node labelled with the concept river ∧ rivière, which is a formal concept that did not exist in the original vocabularies. It corresponds to the instances of 'water-flowing entities' that, although big, flow into other rivers, like Ohio and Saône.

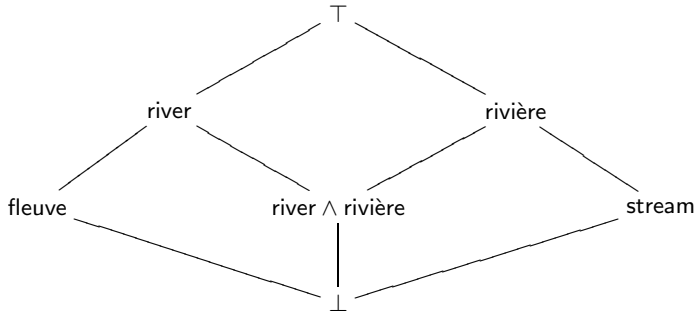


Fig. 5. Concept lattice

3.3 Channel Theory

Channel theory provides the required mathematical machinery to describe the information flow between communities in terms of their connection by means of tokens and types. In order to specify the regularities of each component in a distributed systems channel theory uses the notion of a *local logic*. Separate interacting components will typically use different vocabularies, i.e., they will use different systems of *types*, and also *tokens* and of how these *classified* according to the types.

In addition, each community will have its own particular *constraints* that describe the local behaviour of their instances with respect to their system of types. A *local logic* brings all these ideas together:

Definition 1. *A* local logic *is a quadruple* $\mathcal{L} = (I, T, \models, \vdash)$*, where*

1. I *is a set of* instances *(also called* tokens*);*
2. T *is a set of* types*;*
3. $\models$ *is a* classification relation*, a binary relation between elements of* I *and* T*;*
4. $\vdash$ *is a* consequence relation*, a binary relation between subsets of* T*;*

There are two parts of a local logic that are of particular importance in the channel-theory framework. The first one is the triple $(I, T, \models)$, and is called the *classification* of the local logic, because the binary relation $\models$ determines a classification of instances in I with respect to types in T. Thus, $x \models a$ means that instance $x \in I$ is classified as of type $a \in T$.

The second important part is the pair $(T, \vdash)$, which is called the *theory* of the local logic. This theory is specified by a set of *sequents* $\langle \Gamma, \Delta \rangle$, i.e., pairs where $\Gamma, \Delta \subseteq T$. The set of types Γ is to be interpreted conjunctively, the set Δ disjunctively, so that an instance $x \in I$ *satisfies* a sequent $\langle \Gamma, \Delta \rangle$ provided that, if x is of *every* type in Γ, then x is of *some* type in Δ. Sequents that belong to the theory of a logic are called *constraints* and denoted $\Gamma \vdash \Delta$. Theories of local logics must satisfy the following conditions of *regularity*:[4]

1. Identity: $a \vdash a$, for all $a \in T$;
2. Weakening: If $\Gamma \vdash \Delta$ then $\Gamma, \Gamma' \vdash \Delta, \Delta'$, for all $\Gamma, \Gamma', \Delta, \Delta' \subseteq T$;
3. Global Cut: If $\Gamma, T'_0 \vdash \Delta, T'_1$ for each partition[5] $\langle T'_0, T'_1 \rangle$ of any $T' \subseteq T$, then $\Gamma \vdash \Delta$, for all $\Gamma, \Delta \subseteq T$.

There is an additional element in local logics that we have deliberately left out in Definition 1. Ideally, instances of a local logic adhere to its constraints, although, we cannot presuppose this in general, and exceptions may occur. Local logics also distinguish a subset $N \subseteq I$ of *normal instances* that must satisfy all constraints of the local logic. The idea of a normal instance is needed if we want to model reasonable but unsound flow of information. For the purposes of IF-Map, though, we shall assume that all instances are normal. Such logics are said to be *sound*.

For information to flow between separate components of a distributed system, we need to link local logics that characterise components in a sensible way. This will essentially affect the system of classifications and its associated theory, but in a way that allows the information to flow. This latter is captured with the idea of a *logic infomorphism*:

Definition 2. *A* logic infomorphism $f : \mathcal{L} \rightleftarrows \mathcal{L}'$ *from local logic* $\mathcal{L} = (I, T, \models, \vdash)$ *to local logic* $\mathcal{L}' = (I', T', \models', \vdash')$ *is a contra-variant pair of functions* $f = \langle f^\star, f_\star \rangle$*, where* $f^\star : T \to T'$ *and* $f_\star : I' \to I$*, such that,*

[4] Regularity arises from the observation that, given a classification of instances to types, the set of all sequents that are satisfied by all instances do fulfil these properties.

[5] A partition of T' is a pair $\langle T'_0, T'_1 \rangle$ of subsets of T', such that $T'_0 \cup T'_1 = T'$ and $T'_0 \cap T'_1 = \emptyset$; T'_0 and T'_1 may themselves be empty (hence they form actually a quasi-partition).

1. *for all* $x \in I'$ *and* $a \in T$, $f_\star(x) \models a$ *if and only if* $x \models' f^\star(a)$;
2. *for all* $\Gamma, \Delta \subseteq T$, *if* $\Gamma \vdash \Delta$, *then* $f^\star[\Gamma] \vdash' f^\star[\Delta]$.[6]

The restriction of a logic infomorphism to the classification part of local logics is called only an *infomorphism*.

3.4 Ontologies and Ontology Morphisms

For the purposes of IF-Map described in this paper, we adopt a definition of ontology that includes some of the core components that are usually part of an ontology: *Concepts*, organised in an *is-a hierarchy*, which we capture with a partial-order relation '$\leqslant$'; *relations* defined over these concepts; and notions of *disjointness* of two concepts—when no instance can be considered of both concepts—and *coverage* of two concept—when all instances are covered by two concepts.[7]

Disjointness and coverage are typically specified by means of ontological axioms. IF-Map takes these kind of axioms into account including disjointness and coverage into the hierarchy of concepts by means of two binary relations '$\perp$' and '$|$', respectively.

Definition 3. *An* ontology *is a tuple* $\mathcal{O} = (C, R, \leqslant, \perp, |, \sigma)$ *where*

1. C *is a finite set of concept symbols;*
2. R *is a finite set of relation symbols;*
3. $\leqslant$ *is a reflexive, transitive and anti-symmetric relation on* C *(a partial order);*
4. $\perp$ *is a symmetric and irreflexive relation on* C *(disjointness);*
5. $|$ *is a symmetric relation on* C *(coverage); and*
6. $\sigma : R \to C^+$ *is the function assigning to each relation symbol its arity; the functor* $(-)^+$ *sends a set* C *to the set of finite tuples whose elements are in* C.

When discarding binary relations $\perp$ and $|$, this definition is equivalent to that of a *core ontology* in [41].

When an ontology $\mathcal{O} = (C, R, \leqslant, \perp, |, \sigma)$ is used in some particular application domain, we need to populate it with instances. First, we will have to classify objects of a set X according to the concept symbols in C by defining a binary classification relation $\models_{\mathbf{C}}$. This will determine a classification $\mathbf{C} = (X, C, \models_{\mathbf{C}})$. Next, we will have to specify over which instances the relations represented by the symbols in R are to hold, thus classifying finite tuples of objects of X to the relation symbols in R by defining a binary classification relation $\models_{\mathbf{R}}$. This will determine a classification $\mathbf{R} = (X^+, R, \models_{\mathbf{R}})$. Both classifications will have to be defined in such a way that the partial order $\leqslant$, the disjointness $\perp$, the coverage $|$, and the arity function σ are respected:

[6] $f^\star[\Gamma]$ and $f^\star[\Delta]$ denote the set images of sets Γ and Δ along function $f^\star$, respectively.

[7] Both disjointness and coverage can easily be extended to more than two concepts, although we stay with binary relations, for the ease of presentation.

Definition 4. *A* populated ontology *is a tuple* $\widetilde{\mathcal{O}} = (\mathbf{C}, \mathbf{R}, \leqslant, \perp, |, \sigma)$ *such that* $\mathbf{C} = (X, C, \models_{\mathbf{C}})$ *and* $\mathbf{R} = (X^+, R, \models_{\mathbf{R}})$ *are classifications and* $\mathcal{O} = (C, R, \leqslant, \perp, |, \sigma)$ *is an ontology; we say the ontology is* correct *when, for all* $x, x_1, \ldots, x_n \in X$, $c, d \in C$, $r \in R$, *and* $\sigma(r) = \langle c_1, \ldots, c_n \rangle$,

1. *if* $x \models_{\mathbf{C}} c$ *and* $c \leqslant d$, *then* $x \models_{\mathbf{C}} d$;
2. *if* $x \models_{\mathbf{C}} c$ *and* $c \perp d$, *then* $x \not\models_{\mathbf{C}} d$;
3. *if* $c \mid d$, *then* $x \models_{\mathbf{C}} c$ *or* $x \models_{\mathbf{C}} d$;
4. *if* $\langle x_1, \ldots, x_n \rangle \models_{\mathbf{R}} r$ *then* $x_i \models_{\mathbf{R}} c_i$, *for all* $i = 1, \ldots, n$.

Notice that we write $\widetilde{\mathcal{O}}$ for a populated ontology and $\mathcal{O}$ for the respective unpopulated one.

Transformations of mathematical structures that preserve the structure that characterises them are usually described with homomorphism (or morphisms, for short). Thus, we study the mapping of ontologies through the morphisms of those mathematical structures we have defined for ontologies in Definition 3. The concept of 'populated ontology' is central to our approach to ontology mapping, and we shall use it later in Proposition 1 in order to justify the following definition of an ontology morphism:

Definition 5. *Given two ontologies* $\mathcal{O} = (C, R, \leqslant, \perp, |, \sigma)$ *and* $\mathcal{O}' = (C', R', \leqslant', \perp', |', \sigma')$, *an* ontology morphism $\langle f^\star, g^\star \rangle : \mathcal{O} \to \mathcal{O}'$ *is a pair of functions* $f^\star : C \to C'$ *and* $g^\star : R \to R'$, *such that, for all* $c, d \in C$, $r \in R$, *and* $\sigma(r) = \langle c_1, \ldots, c_n \rangle$,

1. *if* $c \leqslant d$, *then* $f^\star(c) \leqslant' f^\star(d)$;
2. *if* $c \perp d$, *then* $f^\star(c) \perp' f^\star(d)$;
3. *if* $c \mid d$, *then* $f^\star(c) \mid' f^\star(d)$;
4. *if* $\sigma'(g^\star(r)) = \langle c'_1, \ldots, c'_n \rangle$, *then* $c'_i \leqslant' f^\star(c_i)$, *for all* $i = 1, \ldots, n$.

3.5 Information Flow between Ontologies

Our approach to ontology mapping is built upon the assumption that, in the context of channel theory, local logics characterise ontologies.

Hence, a populated ontology $\widetilde{\mathcal{O}} = (\mathbf{C}, \mathbf{R}, \leqslant, \perp, |, \sigma)$—with $\mathbf{C} = (X, C, \models_{\mathbf{C}})$—determines a local logic $\mathcal{L} = (X, C, \models_{\mathbf{C}}, \vdash)$ whose theory $(C, \vdash)$ is given by the smallest regular consequence relation (i.e., the smallest relation closed under Identity, Weakening, and Global Cut) such that, for all $c, d \in C$

$$\begin{array}{lll} c \vdash d & \text{iff} & c \leqslant d \\ c, d \vdash & \text{iff} & c \perp d \\ \vdash c, d & \text{iff} & c \mid d \end{array}$$

The characterisation of an ontology as a local logic justifies the IF-Map method presented in next section, which stems from our intention— explained in Section 2—to map an unpopulated ontology $\mathcal{O} = (C, R, \leqslant, \perp, |, \sigma)$ to a populated one $\widetilde{\mathcal{O}}' = (\mathbf{C}', \mathbf{R}', \leqslant', \perp', |', \sigma')$, by looking at the information flow. For this reason we "artificially" populate the concept types given in C and the relation types

given in R to obtain classifications $\mathbf{C} = (Y, C, \models_{\mathbf{C}})$ and $\mathbf{R} = (Z, R, \models_{\mathbf{R}})$ (unlike a populated ontology, the instances of R need not to be finite tuples of instances of C), and further establish infomorphisms $f : \mathbf{C} \rightleftarrows \mathbf{C}'$ and $g : \mathbf{R} \rightleftarrows \mathbf{R}'$, such that their type-level components $f^\star$ and $g^\star$ constitute an ontology morphism; because in that case we know that the populated ontology $\widetilde{\mathcal{O}}'$ will be a *correct extension* of $\mathcal{O}$, in the sense that the images of $\widetilde{\mathcal{O}}'$'s instances conform to $\mathcal{O}$, as stated in the following proposition:

Proposition 1. *Let $\mathcal{O} = (C, R, \leqslant, \bot, |, \sigma)$ be an (unpopulated) ontology, and let $\widetilde{\mathcal{O}}' = (\mathbf{C}', \mathbf{R}', \leqslant', \bot', |', \sigma')$ be a populated ontology with classifications $\mathbf{C}' = (X', C', \models_{\mathbf{C}'})$, $\mathbf{R}' = (X'^{+}, R', \models_{\mathbf{R}'})$. Let $\mathbf{C} = (Y, C, \models_{\mathbf{C}})$ and $\mathbf{R} = (Z, R, \models_{\mathbf{R}})$ be two classifications whose types are the concept and relation types of $\mathcal{O}$. If $\widetilde{\mathcal{O}}'$ is correct and $f : \mathbf{C} \rightleftarrows \mathbf{C}'$ and $g : \mathbf{R} \rightleftarrows \mathbf{R}'$ are infomorphisms, such that $(f^\star, g^\star) : \mathcal{O} \to \mathcal{O}'$ is an ontology morphism, then, for all $x, x_1, \ldots, x_n \in X$, $c, d \in C$, $r \in R$, and $\sigma(r) = \langle c_1, \ldots, c_n \rangle$,*

1. *$f_\star(x) \models_{\mathbf{C}} c$ and $c \leqslant d$ imply $f_\star(x) \models_{\mathbf{C}} d$;*
2. *$f_\star(x) \models_{\mathbf{C}} c$ and $c \bot d$ imply $f_\star(x) \not\models_{\mathbf{C}} d$;*
3. *$c \mid d$ implies $f_\star(x) \models_{\mathbf{C}} c$ or $f_\star(x) \models_{\mathbf{C}} d$;*
4. *$g_\star(\langle x_1, \ldots, x_n \rangle) \models r$ implies $f_\star(x_i) \models c_i$, for all $i = 1, \ldots, n$.*

Proof.

1. Suppose $f_\star(x) \models_{\mathbf{C}} c$ and $c \leqslant d$. Since f is an infomorphism, $x \models_{\mathbf{C}'} f^\star(c)$. Furthermore, $c \leqslant d$ implies $f^\star(c) \leqslant' f(d)$ because $\langle f^\star, g^\star \rangle$ is an ontology morphism; consequently, $x \models_{\mathbf{C}'} f^\star(d)$. Finally, since f is a infomorphism, $f_\star(x) \models_{\mathbf{C}} d$.
2. Analogous to 1.
3. Analogous to 1.
4. Suppose $g_\star(\langle x_1, \ldots, x_n \rangle) \models r$. Because g is an infomorphism, $\langle x_1, \ldots, x_n \rangle \models g^\star(r)$. Let $\sigma(g^\star(r)) = \langle c'_1, \ldots, c'_n \rangle$. By the correctness of $\widetilde{\mathcal{O}}'$, $x_i \models c'_i$, for all $i = 1, \ldots, n$, and since $\langle f^\star, g^\star \rangle$ is an ontology morphism, $x_i \models f^\star(c_i)$, for all $i = 1, \ldots, n$. Consequently, and because f is a infomorphism, $f_\star(x_i) \models c_i$, for all $i = 1, \ldots, n$.

In the next section we describe the ontology mapping method based on the above characterisation of ontologies as local logics, and ontology morphisms as logic infomorphisms.

4 The IF-Map Method

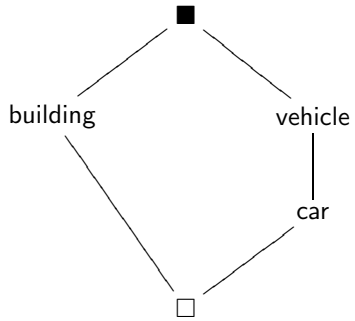
We propose a method for mapping ontologies that draws on the mathematical foundations of information-flow, and we shall use a small easy-to-follow example to illustrate the core parts of IF-Map.

4.1 Reference and Local Ontology

Let us assume that we want to map two ontologies, a reference ontology with a local ontology. We follow the scenario given in Section 2 and assume that the reference ontology has no instances defined, just concept types and constraints over those types. The local ontology, however, has instances classified under its concept types according to a classification relation.

Let the reference ontology be $\mathcal{O} = (C, R, \leqslant, \bot, |, \sigma)$, with

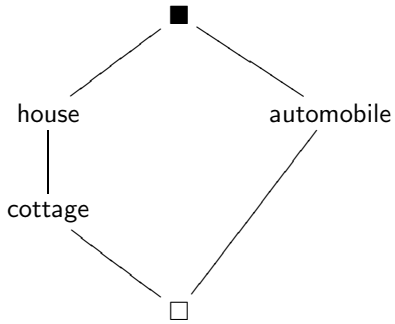
- concepts $C = \{\mathsf{building}, \mathsf{vehicle}, \mathsf{car}\}$;
- relations $R = \{\mathsf{hasParkingSpaceFor}\}$;
- arities $\sigma(\mathsf{hasParkingSpaceFor}) = \langle \mathsf{building}, \mathsf{vehicle} \rangle$; and
- partial order $\leqslant$, disjointness $\bot$, and coverage $|$ as defined by the following lattice:



where ■ is the top and □ is the bottom of the lattice, i.e., $\mathsf{building} \mid \mathsf{vehicle}$ and $\mathsf{building} \bot \mathsf{vehicle}$.

Let the local ontology be $\mathcal{O}' = (C', R', \leqslant', \bot', |', \sigma')$, with

- concepts $C' = \{\mathsf{house}, \mathsf{cottage}, \mathsf{automobile}\}$;
- relations $R' = \{\mathsf{hasGarageFor}, \mathsf{hasShelterFor}\}$;
- arities $\sigma'(\mathsf{hasGarageFor}) = \langle \mathsf{house}, \mathsf{automobile} \rangle$, $\sigma'(\mathsf{hasShelterFor}) = \langle \mathsf{cottage}, \mathsf{automobile} \rangle$; and
- partial order $\leqslant'$, disjointness $\bot'$, and coverage $|'$ as defined by the following lattice:



where ■ is the top and □ is the bottom of the lattice, i.e., $\mathsf{house} \mid' \mathsf{automobile}$ and $\mathsf{house} \bot' \mathsf{automobile}$.

The local ontology, unlike reference ontology, is populated with instances $X = \{cabrio, my\text{-}home, 4wd, new\text{-}inn, old\text{-}inn, coupe\}$, which are classified as follows,

$\models_{\mathbf{C}'}$	house	cottage	automobile
cabrio	0	0	1
my-home	1	0	0
4wd	0	0	1
new-inn	1	1	0
old-inn	1	1	0
coupe	0	0	1

This table contains the following information: *cabrio, 4wd* and *coupe* are automobiles, *my-home* is a house, *new-inn* and *old-inn* are specific kinds of houses, namely cottages. It specifies the classification $\mathbf{C}' = (X, C', \models_{\mathbf{C}'})$.

4.2 Characterisation as Local Logics

In order to automatically find mappings between the reference and the local ontologies that conform to the definition of ontology morphism given in Definition 5, we will need to look for logic infomorphisms between the local logics that characterise these ontologies. First we shall concentrate on the concepts symbols and leave the relation symbols for Section 4.4.

The reference ontology, which is not populated, is characterised by the following local logic. Its regular theory $(C, \vdash)$ has concept symbols as types, and $\vdash$ is the smallest consequence relation closed by Identity, Weakening, and Global Cut that includes the following constraints:

$$\begin{aligned} \mathsf{building,vehicle} &\vdash \\ \mathsf{car} &\vdash \mathsf{vehicle} \\ &\vdash \mathsf{building,vehicle} \end{aligned}$$

Recall that the comma on the left-hand side of these constraints has conjunctive force whereas on the right-hand side it has disjunctive force. Following this, we can give a declarative reading of the above constraints: Nothing is both a building and a vehicle; all cars are vehicles; and everything is either a building or a vehicle.

Since the reference ontology does not have instances of its own, we will need to provide the theory with a set of instances and a classification of these instances with respect to the types. This can be achieved due to a Fundamental Representation Theorem (see [3]), which states that a local logic that is generated from the structure given in a classification is equivalent to the local logic constructed from its theory by generating formally created instances as follows:

1. We take as instances Y all those sequents $\langle \Gamma, \Delta \rangle$ that
   - form a partition of the set of concepts ($\Gamma \cup \Delta = C$ and $\Gamma \cap \Delta = \emptyset$); and
   - are *not* constraints of the theory ($\Gamma \nvdash \Delta$)

For the theory given above, these sequents are

$$\langle\{\mathsf{vehicle,car}\},\{\mathsf{building}\}\rangle$$
$$\langle\{\mathsf{building}\},\{\mathsf{vehicle,car}\}\rangle$$
$$\langle\{\mathsf{vehicle}\},\{\mathsf{building,car}\}\rangle$$

2. We then classify these instances according to the concepts that occur in the left-hand side component of the sequent:

$\models_{\mathbf{C}}$	building	vehicle	car
$\langle\{\mathsf{vehicle,car}\},\{\mathsf{building}\}\rangle$	0	1	1
$\langle\{\mathsf{building}\},\{\mathsf{vehicle,car}\}\rangle$	1	0	0
$\langle\{\mathsf{vehicle}\},\{\mathsf{building,car}\}\rangle$	0	1	0

The generation of instances by means of sequents and their classification to types may not seem obvious, but it is based on the fact that these sequents 'code' the content of the classification table (the left-hand sides of the these sequents indicate which columns of the table bear a '1', while the right-hand sides indicate which columns bear a '0'). The local logic that characterises the reference ontology, i.e., the ontology given by $\mathcal{O}$, is $\mathcal{L} = (Y, C, \models_{\mathbf{C}}, \vdash)$.

The local ontology is populated, and hence has already instances and a classification relation. We only need to derive the theory of the local logic that characterises its concept hierarchy as specified in the lattice above. Therefore, its regular theory $(C', \vdash')$ has concept symbols as types, and $\vdash'$ is the smallest consequence relation closed by Identity, Weakening, and Global Cut that includes the following constraints:

$$\mathsf{house,automobile} \vdash'$$
$$\mathsf{cottage} \vdash' \mathsf{house}$$
$$\vdash' \mathsf{house,automobile}$$

The local logic that characterises the local ontology is, thus, $\mathcal{L}' = (X, C', \models_{\mathbf{C}'}, \vdash')$.

4.3 Generation of Ontology Morphisms via Infomorphisms

To map the ontologies, we must find an ontology morphism from $\mathcal{O}$ to $\mathcal{O}'$, which means that there must exist a logic infomorphism $f = \langle f^{\star}, f_{\star}\rangle$ from local logic $\mathcal{L}$ to local logic $\mathcal{L}'$. This amounts to first look for an infomorphism between their respective classifications:

- A map of concepts $f^{\star} : C \to C'$ (concept-level);
- a map $f_{\star}$ from instances *cabrio*, ..., *coupe* to the formally created instances of the reference ontology (instance-level);

Note that an ontology morphism, as defined in Definition 5, only captures the concept-level of the infomorphism, i.e. $f^\star$. But $f^\star$ has to map the concepts in a way that it respects the hierarchy. One possible way would be:

$$\begin{aligned} f^\star(\mathsf{building}) &= \mathsf{house} \\ f^\star(\mathsf{vehicle}) &= \mathsf{automobile} \\ f^\star(\mathsf{car}) &= \mathsf{automobile} \end{aligned}$$

However, we should point out that the automatic generation of these maps is growing exponentially. But we can use the constraint that the map has to respect the concept hierarchy and limit the number of possible maps. Once the map is fixed, there is at most one acceptable way to map the instances in order for f to be an infomorphism.

We do that by building the following table that represents an infomorphism:[8] We label rows by the instances in $X = \{cabrio, \ldots, coupe\}$ of the local ontology, and columns by the reference ontology's concepts $C = \{\mathsf{building}, \mathsf{vehicle}, \mathsf{car}\}$. We put under each of these concepts the values of the column of the local ontology's classification table that corresponds to the image along the map of ontologies $f^\star$ (i.e., under building we put the column of house):

	building	vehicle	car
cabrio	0	1	1
my-home	1	0	0
4wd	0	1	1
new-inn	1	0	0
old-inn	1	0	0
coupe	0	1	1

Each row should identify (taking into account the classification table of the reference ontology) the formal instances to which each local instance should be mapped onto. Hence, we have the following instance-component of our infomorphism:

$$\begin{aligned} f_\star(cabrio) &= \langle\{\mathsf{vehicle}, \mathsf{car}\}, \{\mathsf{building}\}\rangle \\ f_\star(my\text{-}home) &= \langle\{\mathsf{building}\}, \{\mathsf{vehicle}, \mathsf{car}\}\rangle \\ f_\star(4wd) &= \langle\{\mathsf{vehicle}, \mathsf{car}\}, \{\mathsf{building}\}\rangle \\ f_\star(new\text{-}inn) &= \langle\{\mathsf{building}\}, \{\mathsf{vehicle}, \mathsf{car}\}\rangle \\ f_\star(old\text{-}inn) &= \langle\{\mathsf{building}\}, \{\mathsf{vehicle}, \mathsf{car}\}\rangle \\ f_\star(coupe) &= \langle\{\mathsf{vehicle}, \mathsf{car}\}, \{\mathsf{building}\}\rangle \end{aligned}$$

We can also interpret the above table (and its resulting mapping of instances) as follows: *cabrio* is classified as both a vehicle and a car, according to the reference ontology. No other classification is possible without violating the definition of infomorphism. If *cabrio* was a vehicle but not a car, the local ontology would have been classifying its instances in a way that does not conform to the reference ontology and the fixed map of concepts.

[8] Infomorphisms can themselves be represented by means of classification tables; this draws on theoretical work based on Chu spaces [17, 1, 36].

4.4 Relations and their Arities

In order to constrain the search space when infomorphisms are generated in an automated way, we use ontological relations to guide the classification process that will result in the ontology mapping, namely by looking for infomorphisms $g : \mathbf{R} \rightarrow \mathbf{R}'$ in a similar fashion as before. So, in our example case, we have the following relation defined in the reference ontology:

$$\mathsf{hasParkingSpaceFor} : \mathsf{building} \times \mathsf{vehicle}$$

that is, the binary relation hasParkingSpaceFor holds over building and vehicle. Similarly, in the local ontology we have the following two binary relations:

$$\mathsf{hasGarageFor} : \mathsf{house} \times \mathsf{automobile}$$
$$\mathsf{hasShelterFor} : \mathsf{cottage} \times \mathsf{automobile}$$

These `Local` relations could be used to classify pairs of local instances:

	hasShelterFor	hasGarageFor
$\langle$*my-home*, *cabrio*$\rangle$	0	0
$\langle$*my-home*, *4wd*$\rangle$	0	0
$\langle$*my-home*, *coupe*$\rangle$	1	1
$\langle$*new-inn*, *cabrio*$\rangle$	1	0
$\langle$*new-inn*, *4wd*$\rangle$	0	0
$\langle$*new-inn*, *coupe*$\rangle$	1	0
$\langle$*old-inn*, *cabrio*$\rangle$	0	0
$\langle$*old-inn*, *4wd*$\rangle$	1	0
$\langle$*old-inn*, *coupe*$\rangle$	0	0

That is, my home has a garage (also considered a shelter) only for a coupe, the new inn has a shelter for a cabrio and a coupe, and the old inn has shelter for a 4wd. We then take these pairs and classify them according to the concepts of the reference ontology to determine the mapping of these ontologies:

1. Generate a classification of the above pairs with respect to the reference ontology's relation, by taking any of the two columns of the table above; this gives us two possibilities to explore, suppose we choose:

	hasParkingSpaceFor
$\langle$*my-home*, *cabrio*$\rangle$	0
$\langle$*my-home*, *4wd*$\rangle$	0
$\langle$*my-home*, *coupe*$\rangle$	1
$\langle$*new-inn*, *cabrio*$\rangle$	1
$\langle$*new-inn*, *4wd*$\rangle$	0
$\langle$*new-inn*, *coupe*$\rangle$	1
$\langle$*old-inn*, *cabrio*$\rangle$	0
$\langle$*old-inn*, *4wd*$\rangle$	1
$\langle$*old-inn*, *coupe*$\rangle$	0

   This is the column corresponding to hasShelterFor, and consequently $g^{\star}(\mathsf{hasParkingSpaceFor}) = \mathsf{hasShelterFor}$.

2. The arity of relation hasParkingSpaceFor forces to classify the instances as follows:

	building	vehicle	car
cabrio			1
my-home	1		
4wd			1
new-inn	1		
old-inn	1		
coupe			1

3. Then we need to complete the table according to the definition of infomorphism. This is done as follows: Columns have to correspond to columns of the local ontology's classification table. The only possible completion is:

	building	vehicle	car
cabrio	0	1	
my-home	1	0	
4wd	0	1	
new-inn	1	0	
old-inn	1	0	
coupe	0	1	

Hence,

$$f^\star(\mathsf{building}) = \mathsf{house}$$
$$f^\star(\mathsf{vehicle}) \ = \mathsf{automobile}$$

Rows have to correspond to rows of the reference ontology's classification table. The only possible completion is:

	building	vehicle	car
cabrio	0	1	1
my-home	1	0	0
4wd	0	1	1
new-inn	1	0	0
old-inn	1	0	0
coupe	0	1	1

Hence,

$$f^\star(\mathsf{car}) = \mathsf{automobile} \ ,$$

which completes one possible valid ontology mapping.

The steps described above constitute the core part of the IF-Map method. We complement it with heuristic-based techniques to help us initiate the infomorphism generation.

4.5 Initiating the IF-Map Method

Our definition of ontology morphism (Definition 5) enforces an arity-compatibility check to ensure that the local instances are mapped onto appropriate reference types. When automating this step though, we have to be careful for undesired assignments. These arise when the prospective relations to be mapped share the same types but do not have the same semantics. For instance, assume that the reference ontology has relation hasJobTitle defined over concepts employee and string and the local ontology has relation authoredBy defined over string and employee.[9] The infomorphism generation will map the reference concept employee to the local concept string and the reference concept string to the local concept employee, which will inevitably map the hasJobTitle relation to authoredBy by virtue of sharing the same types.

To tackle this problem we are thinking of two possible ways: (a) We provide a partial map of concepts from one ontology to concepts of the other or (b) classify some instances from the local ontology to their counterparts in the reference ontology. This way we can say that the reference concept employee maps onto the local concept employee and the reference concept string maps onto the local concept string and only this mapping between these types is possible. This will constrain the infomorphism generation and the undesired infomorphisms will not appear. To do this partial mapping automatically we employ a set of heuristics (originally described in [20], pp.95–97, and enhanced for IF-Map). In particular, these heuristics are working on a purely syntactic match fashion but they use the *is-a* hierarchy and type checking to find types that are shared by relations in both ontologies. The algorithm goes like that:

1. Find relation names from both ontologies that are syntactically equivalent (i.e., publishedBy from the reference ontology matches publishedBy from the local ontology);
2. check if their argument types match (since we are dealing with binary relations, both argument types have to match, for instance employee for the reference and ontologies; paper for the reference and local ontologies);
3. use these types to fix a partial map to start the infomorphism generation;
4. if step 2 fails, then traverse the *is-a* hierarchy of types and find syntactically common types that subsume or are subsumed by the common relations' argument types (we traverse the *is-a* hierarchy in both directions: We check for parent and child nodes of the starting node);
5. those that are found syntactically equivalent will be used as in step 3 for partially fixing the initial map of the two ontologies;
6. if step 2 yields only one argument type match, use it and do step 4 for the other argument type.

Note that this algorithm relies on the existence of common relation names in both ontologies. These are syntactically invariant type names. We assume

[9] Note that here the reference and local ontologies are not the same ontologies used in the mapping example.

that since the role of reference ontologies within a community is to favour the sharing of knowledge expressed by means of different local ontologies, many of the names of concepts and relations used to express the reference ontology are syntactically equivalent to the ones used in the local ontologies to express the same (or similar) concepts and relations. If they are not, in which case we have syntactically variant type names, we use another algorithm which makes use of similar heuristics but the main difference is that step 2 above is now the first step and we don't employ step 1 at all. This will allow us to find syntactically variant type names that hold over the same argument types (or their dependants after the taxonomy of types has been traversed). However, this algorithm can yield irrelevant results as the types might be semantically different and not be a simple syntactic variation of the same concept.

In this case, the algorithms described cannot initiate IF-Map and then we turn to the second solution proposed above, which is to let the knowledge engineer classify instances manually. This solution though, requires familiarisation of the engineer with both the reference and local ontologies.

5 Application of IF-Map

We applied the algorithmic steps described in the illustrative example of Section 4 in a large-scale experiment that we conducted in the context of the AKT project (`http://www.aktors.org`). We have not finished our experiments yet, but we have done enough to assess IF-Map. The setting of the experiment is as follows: In the AKT project, five participating universities are contributing their own ontologies representing their own important concepts in the domain of computer science departments in the UK. There is also a reference ontology, AKT Reference, which was built collaboratively by interested participants from all five sites. So, we had to deal with five local ontologies and one reference ontology. The local ontologies were populated whereas the reference ontology was not. That is in-line with the IF-Map scenario we described in Section 4. Furthermore, since local ontologies are maintained locally by each participating site, it is anticipated to face a variety of formalisms and use different tools for ontology design, development, and deployment. IF-Map's architecture (see Section 2) however, allows for different formalisms to be used as input.

One of our case studies was to apply IF-Map to map AKT Reference to Southampton's and Edinburgh's local ontologies, Soton and Edin. These local ontologies are populated with a few thousand instances (ranging from 5k to 18k) and a few hundreds of concepts. There are a few axioms defined, and both have relations. AKT Reference is more compact, it has no instances and approximately 65 concepts with 45 relations. There are a few axioms defined as well. In Figure 6 we include two screenshots, one from Ontolingua-encoded AKT Reference—in particular, the organisations sub-ontology—and the translated version of this fragment in Prolog clauses.[10] As we mentioned earlier (Section 2),

[10] The reader should bear in mind that clauses in the Prolog version denote the typing of the relations and not logical implications.

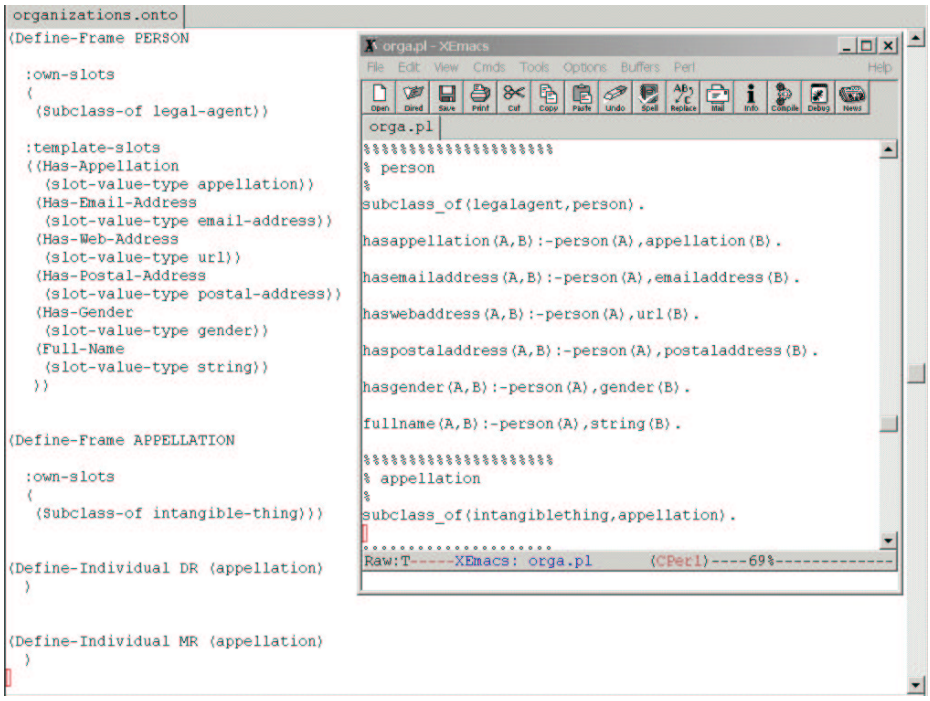


Fig. 6. A window with an Ontolingua fragment of `AKT Reference` overlapped by a smaller window showing a partial translation of this in Prolog clauses.

our translation is a partial and customised for the IF-Map application. Therefore, from the Ontolingua fragment shown in Figure 6 we only extract and transform to Prolog clauses the class hierarchy (by regarding frames as classes) and the relations along with their arities (which is used for type checking to bind appropriate tokens to types when using Prolog's backward chaining inference engine).

In Figure 7 we include two screenshots, one from the Protégé-edited SOTON—in particular, highlighting the class person and its template slots—and the translated version of this fragment in Prolog clauses. As with the reference ontology translation, we partially translate predefined fragments of the local ontology like class hierarchy and relations along with arities.

In Figure 8 we include a screenshot of our Web accessible RDF results page for some relations and concepts. In this page, we show a small fraction of the results from mapping concepts and relations from AKT REFERENCE to to their counterparts in SOTON. The concepts shown can be traced back to the previous figures where the original format is shown. As we can see, apart from mapping concepts, like AKT REFERENCE's document to SOTON's publication we also map relations: AKT REFERENCE's hasappellation to SOTON's title. The arities of these relations allow this sort of mapping, whereas in other ontologies this would

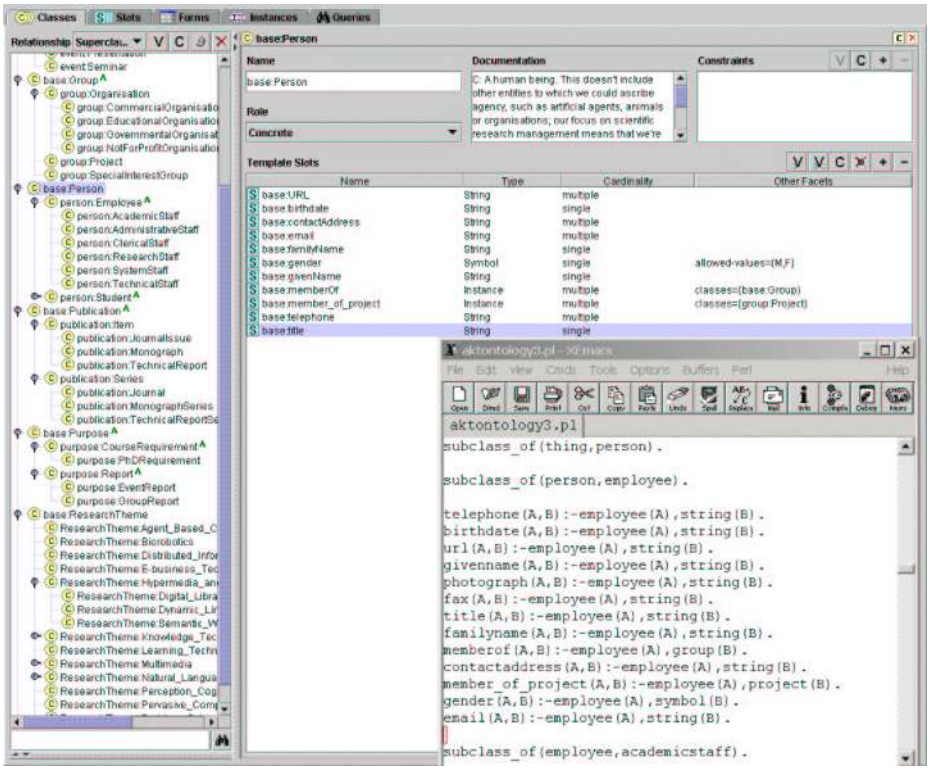


Fig. 7. A window with a Protégé fragment of `Soton` overlapped by a smaller window showing a partial translation of this in Prolog clauses.

have been inappropriate, when for example title refers to title of a paper. These mappings were generated automatically, and IF-Map initiated these with the semantically-rich heuristics we described in 4.5.

The algorithms we have implemented so far are of exponential complexity in the number of concepts, because they are based on a declarative Prolog specification of the mapping principle explained in in Section 4. Consequently, we currently base the IF-Map method on an incremental construction of ontology morphisms, in order to tackle large-scale ontologies: First, only certain manageable fragments of the ontologies are mapped, and next, these fixed maps are used to guide the generation of larger fragments, in the manner explained in Section 4. We are currently investigating heuristics for the automatic identification of such fragments.

Fig. 8. Results of ontology mapping in Web accessible RDF format.

6 Related work

IF-Map, amid its well-defined purpose of ontology mapping and, extensionally, merging, taps on a number of areas and uses techniques discussed in diverse communities. Therefore, it is impossible to compile an exhaustive list of references to related work, but we have deliberately expanded the scope of references to cover as many representative works as possible; for a more in-depth survey on ontology mapping, see [21]. At the same time though, we were careful to identify works that are related somehow with IF-Map's core characteristics: Use of formal definitions of ontology mapping, use of information-flow theory, expressed in a declarative and executable language in a domain and tool independent manner,

applied as a method and as a theory for ontology mapping, and being—under circumstances—fully automatic.

Not all of the references we cite here meet these criteria; some provide features that IF-Map does not support and others focus on a single criterion of the list given above. Nevertheless, the diversity of works reported in this section demonstrates the importance of the topic in a number of communities. This paper's scope, though, prevents us from getting into great detail when describing related work hereinafter, but we give a flavour of the current landscape in ontology mapping research across different communities. Among the few formal approaches in ontology mapping and merging is that of FCA-Merge [41]. It is based on Formal Concept Analysis [14] and it is aimed, mainly, at merging ontologies. Its developers, Stumme and Maedche, incorporate natural language techniques in their FCA-based method to derive a lattice of concepts. The lattice is then explored manually by a knowledge engineer who will build the merged ontology with semi-automatic guidance from FCA-Merge. In particular, FCA-Merge works as follows: The input to the method are a set of documents—from which concepts will be extracted—together with the ontologies that will be merged. These documents should be representative of the domain at question and be related to the ontologies. They also have to cover all concepts from both ontologies as well as separating them well enough. These strong assumptions have to be met in order to obtain good results from the FCA-Merge. As this method relies heavily on the availability of classified instances in ontologies to be merged, the authors argue that this will not be the case in most ontologies so they opt to extract instances from documents. In this respect, the first step of FCA-Merge could be viewed as an ontology population mechanism. This initial step in FCA-Merge could be skipped though, when there is a shared set of instances classified to the concepts in both ontologies. Once the instances will be extracted,[11] Stumme and Maedche construct the concept lattice and from there provide semi-automatic support for the knowledge engineer to derive the final merged ontology.

Formal Concept Analysis has also been used by the database community in their federated databases domain. In particular, Schmitt and Saake employ Formal Concept Analysis techniques to assist database schema integration [37]. The focus of their work is to merge different inheritance hierarchies by decomposing overlapping class extensions into base extensions and use Formal Concept Analysis techniques to inform algorithms for integrating the databases schemata. In the Scalable Knowledge Composition (SKC) project, Jannink et al. [19] presented the use of a rule-based algebra for encapsulating and composing ontologies. Ontologies are clustered in contexts, and the authors use a rule-based algebra to define interfaces to link the extracted contexts with the original ontologies.

Noy and Musen have developed two systems for performing ontology merging and alignment in the Protégé-2000 ontology development environment [16]: SMART [33] and its successor PROMPT [34]. These tools use linguistic similar-

[11] We do not refer to natural language techniques and methods used in this process by FCA-Merge developers, since they are peripheral to our interests in ontology mapping.

ity matches between concepts for initiating the merging or alignment process and then use the underlying ontological structures provided by the Protégé-2000 environment (classes, slots, facets) to inform a set of heuristics for identifying further matches between the ontologies. A similar tool has been developed by McGuinness et al. for the Ontolingua ontology editor: Chimaera [29]. As in PROMPT, this tool is interactive and the engineer is in charge of making decisions that will affect the merging.

From the machine learning perspective we report the works of Lacher and Groh [26] and Doan et al. [9] where their systems employ machine learning algorithms in conjunction with similarity measures to yield prospective mappings between ontology concepts. Other works worth citing here are Chalupsky's OntoMorph [6] translation system for symbolic knowledge, Kiryakov et al.'s OntoMap portal [25] for mapping linguistic ontologies, the OBSERVER system [30] by Mena et al. for information integration, Gangemi et al.'s [13] ONIONS methodology for medical ontologies, Visser and Tamma's heterogeneity categorisation [42], and the reports from Pinto et al. [35] and Noy and Hafner in [32].

7 Conclusion

In this paper we have presented a novel method and a theory for ontology mapping. We formalised the notion of ontology, ontology morphism and ontology mapping and linked them to the formal notions of local logic and logic infomorphism stemming from IF theory. We then applied them in a mechanised manner—IF-Map—to map diverse ontologies. The first results are promising for the application of IF-Map to large-scale ontology mapping efforts and we continue researching fruitful extensions of it, such as ontology merging, reasoning about ontology evolution, and inclusion of ontological axioms.

The automation of ontology mapping as provided by IF-Map provides automated support in the alignment of ontologies by automatically generating mappings between a reference and various local ontologies. For instance, for the special case of data integration, this would correspond to automatic generation of source descriptions, e.g. formalised as views over a mediated schema [18]. The correspondence, though, is not exact, as views over a mediated schema could be more expressive than IF-Maps current concept-to-concept and relation-to-relation mapping.

In this sense it would be interesting to work on the integration of previous approaches to semantic integration by the database community with recent efforts carried out by the ontology community (see, e.g., [21]) in the context of our approach to ontology mapping based on the Barwise-Seligman theory of information flow, as it might further illuminate the long road lying ahead.

Acknowledgements

An earlier version of this paper was presented under the title "Information-Flow based Ontology Mapping" at the First International Conference on Ontologies,

Databases and Applications of Semantics (ODBASE'02), Irvine CA, USA, October 2002. We are grateful to Karl Aberer for selecting an extended version of that paper for this special volume, and we would like to thank the audience of ODBASE'02 and the reviewers for their valuable comments.

This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

Marco Schorlemmer is also supported by a 'Ramón y Cajal' Fellowship from the Spanish Ministry of Science and Technology.

References

1. M. Barr. The Chu construction. *Theory and Applications of Categories*, 2(2):17–35, 1996.
2. J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.
3. J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.
4. C. Batini, M. Lenxerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.
6. H. Chalupsky. OntoMorph: A translation system for symbolic knowledge. In *7th International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, Colorado, USA, Apr. 2000.
7. F. Corrêa da Silva, W. Vasconcelos, D. Robertson, V. Brilhante, A. de Melo, M. Finger, and J. Agustí. On the insufficiency of ontologies: Problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems*, 15(3):147–167, 2002.
8. K. Devlin. *Logic and Information*. Cambridge University Press, 1991.
9. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *11th International World Wide Web Conference*, Honolulu, Hawaii, USA, May 2002.
10. J. Domingue. Tadzebao and WebOnto: Discussing, browsing, and editing ontologies on the web. In *11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, Apr. 1998.
11. F. Dretske. *Knowledge and the Flow of Information*. MIT Press, 1981.
12. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727, 1997.
13. A. Gangemi. Ontology integration: Experiences with medical terminologies. In N. Guarino, editor, *Formal Ontology in Information Systems*, volume 46 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1998.
14. B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, 1999.
15. M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format (KIF). Draft proposed American National Standard, NCITS.T2/98-004, 1998.

16. W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, and M. Musen. Knowledge modeling at the millennium (the design and evolution of Protégé-2000). In *12th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, Oct. 1999.
17. V. Gupta. *Chu Spaces: A Model of Concurrency*. PhD thesis, Stanford University, 1994.
18. A. Halevy. Answering queries using views. *The VLDB Journal*, 10:270–294, 2001.
19. J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. Encapsulation and composition of ontologies. In *AAAI'98 Workshop on Information Integration*, Madison, Wisconsin, USA, July 1998.
20. Y. Kalfoglou. *Deploying Ontologies in Software Design*. PhD thesis, Division of Informatics, The University of Edinburgh, June 2000.
21. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The sate of the art. *Knowledge Engineering Review*, 18(2), 2003.
22. R. Kent. The information flow foundation for conceptual knowledge organization. In *6th International Conference of the International Society for Knowledge Organization*, Toronto, Canada, July 2000.
23. R. Kent. A KIF formalization of the IFF category theory ontology. In *IJCAI 2001 Workshop of the IEEE Standard Upper Ontology*, Seattle, Washington, USA, 2001.
24. R. Kent. The IFF foundation for ontological knowledge organization. In *Knowledge Organization and Classification in International Information Retrieval*, Cataloging and Classification Quarterly. The Haworth Press Inc., 2003.
25. A. Kiryakov, K. Simov, and M. Dimitrov. OntoMap: Portal for upper-level ontologies. In *Second International Conference on Formal Ontology in Information Systems*, Ogunquit, Maine, USA, Oct. 2001.
26. M. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *14th International FLAIRS Conference*, Key West, Florida, USA, May 2001.
27. O. Lassila and R. Swick. Resource description framework (RDF) model and syntax specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, Feb. 1999. W3C Recommendation.
28. B. McBride. Jena: Implementing the RDF model and syntax specification. In *2nd International Workshop on the Semantic Web (SemWeb'2001)*, volume 40 of *CEUR Workshop Proceedings*, Hong Kong, China, May 2001.
29. D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *7th International Conference on Principles of Knowledge Representation and Reasoning*, Breckenridge, Colorado, USA, Apr. 2000.
30. E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain specific ontologies for semantic information brokering on the global information infrastructure. In N. Guarino, editor, *Formal Ontology in Information Systems*, volume 46 of *Frontiers in Artificial Intelligence and Applications*, pages 269–283. IOS Press, 1998.
31. E. Motta. *Reusable Components for Knowledge Modelling: Case Studies in Parametric Design Problem Solving*, volume 53 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1999.
32. N. Noy and C. Hafner. The state of the art in ontology design: A survey and comparative review. *AI Magazine*, 18(3):53–74, 1997.
33. N. Noy and M. Musen. SMART: Automated support for ontology merging and alignment. In *12th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, Oct. 1999.

34. N. Noy and M. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *17th National Conference on Artificial Intelligence (AAAI'00)*, Austin, Texas, USA, July 2000.
35. S. Pinto, A. Gómez-Pérez, and J. Martins. Some issues on ontology integration. In *IJCAI'99 Workshop on Ontologies and Problem-Solving Methods*, volume 18 of *CEUR Workshop Proceedings*, Stockholm, Sweden, Aug. 1999.
36. V. Pratt. The Stone gamut: A coordination of mathematics. In *10th Annual Symposium on Logic in Computer Science*, pages 444–454. IEEE Computer Society Press, 1995.
37. I. Schmitt and G. Saake. Merging inheritance hierarchies for database integration. In *3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98)*. IEEE Computer Society Press, 1998.
38. M. Schorlemmer. Duality in knowledge sharing. In *7th International Symposium on Artificial Intelligence and Mathematics*, Ft. Lauderdale, Florida, USA, 2002.
39. A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
40. J. Sowa. *Knowledge Representation and Reasoning: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.
41. G. Stumme and A. Maedche. FCA-Merge: Bottom-up merging of ontologies. In *17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, Seattle, Washington, USA, Aug. 2001.
42. P. Visser and V. Tamma. An exeprience with ontology-based agent clustering. In *IJCAI'99 Workshop on Ontologies and Problem-Solving Methods*, volume 18 of *CEUR Workshop Proceedings*, Stockholm, Sweden, Aug. 1999.