Information-Flow-Based Ontology Mapping

Yannis Kalfoglou[1] and Marco Schorlemmer[2⋆]

[1] Advanced Knowledge Technologies (AKT)
Department of Electronics and Computer Science
University of Southampton
[2] Advanced Knowledge Technologies (AKT)
Centre for Intelligent Systems and their Applications
The University of Edinburgh

Abstract. As ontologies become ever more important for semantically-rich information exchange and a crucial element for supporting knowledge sharing in a large distributed environment, like the Web, the demand for sharing them increases accordingly. One way of achieving this ambitious goal is to provide mechanised ways for mapping and merging ontologies. This has been the focus of recent research in knowledge engineering. However, we observe a dearth of mapping methods that are based on a strong theoretical ground, are easy to replicate in different settings, and use semantically-rich mechanisms for performing ontology mapping. In this paper, we aim to fill in these gaps with a method we propose for Information-Flow-based ontology mapping. Our method draws on the proven theoretical ground of Information Flow and channel theory, and we provide a systematic and mechanised methodology for deploying it on a distributed environment to perform ontology mapping among a variety of different ontologies. We applied our method at a large-scale experiment of mapping five ontologies modelling Computer Science departments in five UK Universities. We elaborate on a theory for ontology mapping, analyse the mechanised steps of applying it, and assess its ontology mapping results.

1 Introduction

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 merge ontologies where the product of this merge will be, at the very least, the intersection of the two given ontologies. These are the dominant approaches and have been studied and applied in a variety of systems (see, for example, [31]).

There are, however, some drawbacks that prevent engineers benefiting from such systems. Firstly, the assumptions made in making these mappings and performing merging are not always exposed to the community and no technical

⋆ Last names of authors are in alphabetical order.

R. Meersman, Z. Tari (Eds.): CoopIS/DOA/ODBASE 2002, LNCS 2519, pp. 1132–1151, 2002.

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, a.k.a. 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 worked on a method and a theory for ontology mapping and merging. We were determined to tackle these drawbacks so our approach draws heavily on a proven theoretical ground but at the same time we are providing a systematic approach for ontology mapping and mechanised methodological steps. In particular, in this paper we propose an Information-Flow-based method for ontology mapping (hereafter, IF-Map). We are mostly interested in mapping ontologies, but we can extend the approach to merge them, too. IF-Map draws on the works of Schorlemmer [34] on using Information Flow (hereafter, IF) theory to align ontologies and the heuristics defined by Kalfoglou (in [21], pp.95–97), to analyse prospective mappings between ontologies. On the theoretical side, our method draws on the IF theory described by Barwise and Seligman [3] and the work of Kent on the IF Framework [23] for the IEEE standardisation activity and his proposed methodology for merging ontologies [22]. The methodological part of IF-Map has also been influenced by the work of Stumme and Maedche on the FCA-Merge method [35].

We describe a scenario for ontology mapping and the architecture we built to perform ontology mapping in Section 2. We briefly provide mathematical preliminaries on IF 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 do an evaluation of our method and elaborate on scalability issues. We discuss related work in Section 6 and summerise the paper in Section 7.

2 An Architecture for Ontology Mapping

In Figure 1 we illustrate our approach to ontology mapping. In particular, the focus is on the use of IF as the underpinning mathematical foundation for establishing mappings between two ontologies. We shall formalise these mappings in terms of *logic infomorphisms*, which we introduce in Section 3. Actually, this figure clearly resembles Kent's proposed two-step process in ontology sharing [22], but it has differences in its implementation. The solid rectangular line surrounding `Reference ontology`, `Local ontology 1` and `Local ontology 2` denotes the existing ontologies. We assume that `Local ontology 1` and `Local ontology 2` are ontologies used by different communities and populated with their instances, while `Reference ontology` is an agreed understanding for favouring knowledge sharing, and is not supposed to be populated. The dashed rectangular line sur-

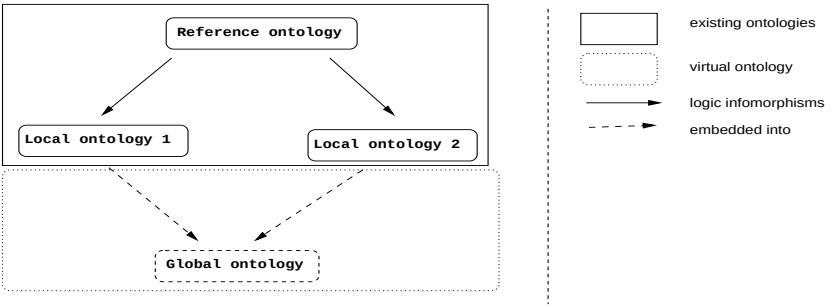


Fig. 1. The scenario for ontology mapping.

rounding `Global ontology` denotes an ontology that does not exist yet, but will be constructed "on the fly" for the purpose of alignment. This is similar to Kent's "virtual ontology of community connections" [22]. The solid arrow lines linking `Reference ontology` with `Local ontology 1` and `Local ontology 2` denote IF between these ontologies formalised as logic informorphisms. In this paper we present the methodology to generate these logic infomorphisms. The dashed arrow lines denote the embedding from `Local ontology 1` and `Local ontology 2` into `Global ontology`. This latter is the sum of the local ontologies *modulo* `Reference ontology` and the generated logic infomorphisms. As we mentioned earlier, this extension would be the merging part of IF-Map.

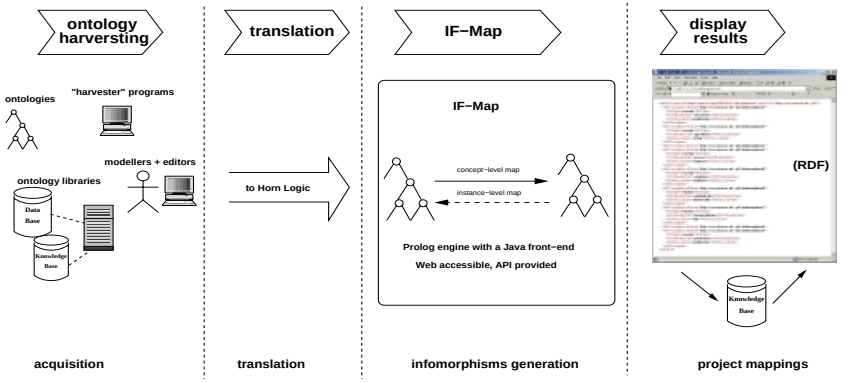


Fig. 2. The IF-Map architecture.

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) infomorphisms, and (d) display results. In the ontology harvesting step we perform our acquisition. We acquire ontologies by using a variety of methods: use existing ontologies, download them from ontology libraries (for example,

from Ontolingua [11] or WebOnto [9] servers), edit them in ontology editors (for example, in Protégé [18]), or harvest 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 for 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-language formats ranging from KIF [17] and Ontolingua to OCML [30], RDF [26], Prolog, and native Protégé knowledge bases.

This introduces the second step in our process, that of translation. As we have declaratively specified the IF-Map method in Horn logic and execute it with the aim of a Prolog engine, we partially translate the above formats to Prolog clauses. Our translator programs are either written in-house, or whenever available, use public translators. For example, there are public RDF to Prolog translators[1] as well as Ontolingua to Prolog. In most of the cases though, we found it practical to write our own translators. 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 [6], 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[2]) 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 knotty problem, and recent research from Corrêa da Silva and colleagues [6] 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 with the aim of an intermediary Java layer, where RDF is being produced using the Jena RDF API [27]. Finally, we store the results in a knowledge base for future reference and maintenance.

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

[2] We briefly describe the principles we used to partially translate from Ontolingua to Prolog in [21], pp.105–107.

Before proceeding to an example case of deploying this architecture we shall introduce the theoretical background of IF-Map. In the next section We expand on channel theory and logic informorphisms, and give a formal account of ontology mapping.

3 Theoretical Preliminaries

In order to give an formal characterisation of ontology mapping we start from the assumption that mapping ontologies presupposes flow of information, and that we need to base any formal notion of ontology mapping on a sound mathematical theory of information and information flow. There is no such theory of information yet, but there have been several efforts in establishing one [10,2,7, 3].

3.1 Channel Theory

Channel theory has been developed 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; furthermore it is the particular instances that carry information, so that information flow crucially involves both types (i.e., the terminology to describe components) and instances.

Central to channel theory is the idea of a *local logic*. Separate interacting communities will typically use different vocabularies, i.e., they will use different systems of *types*, and the *instances* that these communities manage will be *classified* according to these types in quite different ways. 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*;*
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*[3]:

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[4] $\langle T'_0, T'_1 \rangle$ 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 pressupose 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 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 contravariant pair of functions* $f = \langle f^\star, f_\star \rangle$, *where* $f^\star : T \to T'$ *and* $f_\star : I' \to I$, *such that,*

1. *for* $x \in I'$ *and* $a \in T$, $f_\star(x) \models a$ *if and only if* $x \models' f^\star(a)$;
2. *for* $\Gamma, \Delta \subseteq T$, *if* $\Gamma \vdash \Delta$, *then* $f^\star[\Gamma] \vdash' f^\star[\Delta]$[5].

The restriction of logic infomorphisms to the classification part of local logics are called *infomorphisms*.

3.2 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* of an *is-a hierarchy*, which we caputre with a partial order relation '$\leqslant$'; *relations* defined over these concepts; and notions of *disjointness* of

[3] 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 fulfill these properties.

[4] 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 it is actually a quasi-partition).

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

two concepts — when no instance can be considered of both concepts — and *coverage* of two concept — when all instances are covered by two concepts.[6]

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. In a future, we plan do extend IF-Map to cope with full first-order axioms.

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 antisymmetric relation on* C *(a partial order);*
4. $\perp$ *is a symmetric and irreflexive relation on* C*;*
5. $|$ *is a symmetric relation on* C*; 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 [35].

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:

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* sound *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.

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

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* $\mathcal{O} = (C, R, \leqslant, \bot, |, \sigma)$ *and* $\mathcal{O}' = (C', R', \leqslant', \bot', |', \sigma')$, *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* $c \bot d$, *then* $f^\star(c) \bot' f^\star(d)$;
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* $i = 1, \ldots, n$.

3.3 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 $\widetilde{\mathcal{O}} = (\mathbf{C}, \mathbf{R}, \leqslant, \bot, |, \sigma)$ 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 $c, d \in C$

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

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 $\mathcal{O} = (C, R, \leqslant, \bot, |, \sigma)$ to a populated one $\widetilde{\mathcal{O}}' = (\mathbf{C}', \mathbf{R}', \leqslant', \bot', |', \sigma')$, by looking at the information flow. For this reason we "formally" 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 $\mathbf{R} = (Z, R, \models_{\mathbf{R}})$ (notice that, 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 *sound 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 proposistion:

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 sound and $f : \mathbf{C} \rightleftarrows \mathbf{C}'$ and $g : \mathbf{R} \rightleftarrows \mathbf{R}'$ are informorphisms, 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 $c \perp d$ 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 soundness 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

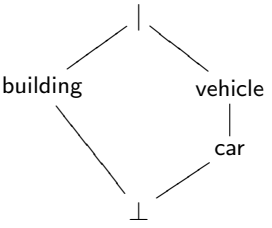
We propose a method for mapping ontologies that draws on the mathematical foundations of information-flow, and we shall use a small easy-to-follow example to illustrate the core parts of IF-Map.

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.

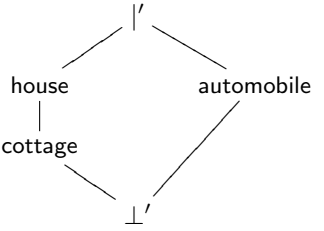
Let `Reference` be the ontology $\mathcal{O} = (C, R, \leqslant, \bot, |, \sigma)$, with

- 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 given lattice.



Let the `Local` be the ontology $\mathcal{O}' = (C', R', \leqslant', \bot', |', \sigma')$, 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 given lattice.



`Local`, unlike `Reference`, is populated with instances $X = \{cabrio, bcn, 4wd, mall, skye, coupe\}$, which are classified as follows,

$\models_{\mathbf{C}'}$	house	cottage	automobile
cabrio	0	0	1
bcn	1	0	0
4wd	0	0	1
mall	1	1	0
skye	1	1	0
coupe	0	0	1

This table contains the following information: *cabrio, 4wd* and *coupe* are automobiles, *bcn* is a house in Barcelona, *mall* and *skye* are specific kinds of house, cottages in Mallorca and the Isle of Skye, respectively. It specifies the classification $\mathbf{C}' = (X, C', \models_{\mathbf{C}'})$.

4.2 Characterisation as Local Logics

In order to automatically find mappings between `Reference` and `Local` 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.

`Reference`, 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:

$$\mathsf{building,vehicle} \vdash \emptyset \qquad \mathsf{car} \vdash \mathsf{vehicle} \qquad \emptyset \vdash \mathsf{building,vehicle}$$

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.

We will need to provide the theory with a set of instances and a classification of these instances with respect to the types. Now, every regular theory determines a classification 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 $\langle \{\mathsf{vehicle,car}\}, \{\mathsf{building}\} \rangle$, $\langle \{\mathsf{building}\}, \{\mathsf{vehicle,car}\} \rangle$, and $\langle \{\mathsf{vehicle}\}, \{\mathsf{building,car}\} \rangle$.
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 local logic that characterises `Reference`, i.e., the ontology given by $\mathcal{O}$, is $\mathcal{L} = (Y, C, \models_{\mathbf{C}}, \vdash)$. The generation of instances by means of sequents and their classification may not seem obvious, but it turns out that classifications generated in this way satisfy a fundamental Representation Theorem (see [3]) stating that a local logic that is generated from the structure given in a classification is equivalent to the local logic constructed from its theory as described above.

`Local` 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:

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

The local logic that characterises `Local` 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:

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

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[7]: we label rows by the instances in $X = \{cabrio, \dots, coupe\}$ of `Local`, and columns by `Reference`'s concepts $C = \{\mathsf{building,vehicle,car}\}$. We put under each of these concepts the values of the column of `Local`'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
bcn	1	0	0
4wd	0	1	1
mall	1	0	0
skye	1	0	0
coupe	0	1	1

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

$$\begin{aligned}
f_\star(cabrio) &= \langle\{\mathsf{vehilce,car}\}, \{\mathsf{building}\}\rangle \\
f_\star(bcn) &= \langle\{\mathsf{building}\}, \{\mathsf{vehilce,car}\}\rangle \\
f_\star(4wd) &= \langle\{\mathsf{vehilce,car}\}, \{\mathsf{building}\}\rangle \\
f_\star(mall) &= \langle\{\mathsf{building}\}, \{\mathsf{vehilce,car}\}\rangle \\
f_\star(skye) &= \langle\{\mathsf{building}\}, \{\mathsf{vehilce,car}\}\rangle \\
f_\star(coupe) &= \langle\{\mathsf{vehilce,car}\}, \{\mathsf{building}\}\rangle
\end{aligned}$$

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

[7] Infomorphisms can themselves be represented by means of classification tables; this draws on theoretical work based on Chu spaces [19,1,32].

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 `Reference`:

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

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

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

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

	hasShelterFor	hasGarageFor
$\langle bcn, cabrio \rangle$	0	0
$\langle bcn, 4wd \rangle$	0	0
$\langle bcn, coupe \rangle$	1	1
$\langle mall, cabrio \rangle$	1	0
$\langle mall, 4wd \rangle$	0	0
$\langle mall, coupe \rangle$	1	0
$\langle skye, cabrio \rangle$	0	0
$\langle skye, 4wd \rangle$	1	0
$\langle skye, coupe \rangle$	0	0

That is, the house in Barcelona has a garage (also considered a shelter) only for a coupe, the cottage in Mallorca has a shelter for a cabrio and a coupe, and the cottage in the Isle of Skye has shelter for a 4wd. We then take these pairs and classify them according to the concepts of `Reference` to determine the mapping of these ontologies:

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

	hasParkingSpaceFor
$\langle bcn, cabrio \rangle$	0
$\langle bcn, 4wd \rangle$	0
$\langle bcn, coupe \rangle$	1
$\langle mall, cabrio \rangle$	1
$\langle mall, 4wd \rangle$	0
$\langle mall, coupe \rangle$	1
$\langle skye, cabrio \rangle$	0
$\langle skye, 4wd \rangle$	1
$\langle skye, coupe \rangle$	0

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

2. The arity of relation hasParkingSpaceFor forces to classify the instances as in Figure 3 (a).
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 `Local`'s classification table. The only possible completion is shown in Figure 3 (b).
   Hence, $f^{\star}(\mathsf{building}) = \mathsf{house}$ and $f^{\star}(\mathsf{vehicle}) = \mathsf{automobile}$. Rows have to correspond to rows of `Reference`'s classification table. The only possible completion is shown in Figure 3 (c).

	building	vehicle	car
cabrio			1
bcn	1		
4wd			1
mall	1		
skye	1		
coupe			1

(*a*)

	building	vehicle	car
cabrio	0		1
bcn	1		0
4wd	0		1
mall	1		0
skye	1		0
coupe	0		1

(*b*)

	building	vehicle	car
cabrio	0	1	1
bcn	1	0	0
4wd	0	1	1
mall	1	0	0
skye	1	0	0
coupe	0	1	1

(*c*)

Fig. 3. Completing the classification table.

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 kick-start the infomorphism generation.

4.5 Kick-Start for 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 `Reference` has relation hasJobTitle defined over concepts employee and string and `Local` has relation authoredBy defined over string and employee[8]. The infomorphism generation will map `Local`'s employee to `Reference`'s string and `Local`'s string to `Reference`'s employee, which will inevitably map 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 representative instances from the `Local` to their `Reference` counterparts.

[8] Note that here `Reference` and `Local` do not denote the same ontologies used in the mapping example.

This way, we can say that Reference's employee maps onto Local's employee and Reference's string maps onto string and only this mapping between these types is possible. This will constrain the infomorphism generation and the offending infomorphisms will not appear. To do this partial mapping automatically we employ a set of heuristics (originally described in [21], pp.95–97). 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 Reference matches publishedBy from Local);
2. check if their argument types match (since we are dealing with binary relations, both argument types have to match, for instance employee for Reference and Local; paper for Reference and Local);
3. use these types to fix a partial map to start the infomorphism generation;
4. if 2 fails, then use the *is-a* hierarchy to traverse the 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 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 4 for the other argument type;

Note that this algorithm relies on the existence of common relation names in both ontologies. This is based on the assumption 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.

In case this fails, the algorithm cannot be initiated and then we turn to the second solution proposed above, which is to let the knowledge engineer classify representative instances manually. This solution though, requires familiarisation of the engineer with both the reference and local ontologies.

5 Evaluation 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. 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 a reference ontology. The local ontologies were populated whereas the reference ontology was not. That is in-line with the IF-Map scenario as we described in Section 4. Furthermore, since local ontologies are maintained locally, by five different sites, it anticipated to use a variety of formalisms and tools for ontology design, development, and deployment. IF-Map's architecture (see Section 2) allows for different formalisms to be used as input.

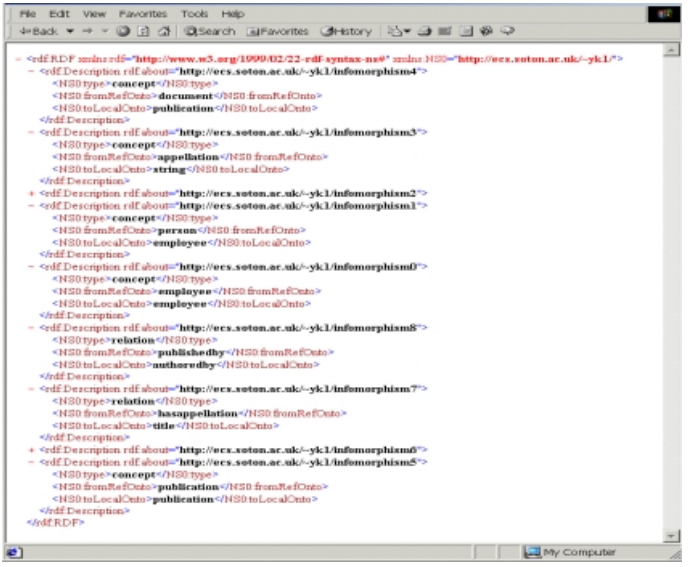


Fig. 4. Results of ontology mapping in Web accessible RDF format.

We applied IF-Map to map `AKT Reference` to Southampton's and Edinburgh's local ontologies. 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. The `AKT Reference` ontology is more compact, it has no instances and approximately 65 concepts with 45 relations. There are a few axioms defined as well. In Figure 4 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 Southampton's ontology. As we can see, apart from mapping concepts, like `AKT Reference`'s document to Southampton's publication we also map relations: `AKT Reference`'s hasappellation to Southampton's title. The arities of these relations allow this sort of mapping, whereas in other ontologies this would have been inappropriate, when for example title refers to title of a paper. These mappings were generated automatically; IF-Map initiated these experiments 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. Although the implementation of IF-Map as it currently stands can still be improved by using more sophisticated algorithms for ontology morphism generation, we are basing 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.

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. 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. Space reasons and this paper's scope prevents us from getting into great detail when describing related work hereinafter, but we aim to 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 [35]. It is based on Formal Concept Analysis [16] and it is aimed, mainly, at merging ontologies, hence, FCA-Merge. 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 and 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. Once the concepts will be extracted, the authors construct the concepts lattice and from there provide semi-automatic support for knowledge engineers 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 [33]. 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 and colleagues [20] 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.

Fridman Noy and Musen have developed two systems for performing ontology merging and alignment in the Protégé-2000 [18] ontology development environment: SMART [13] and its successor PROMPT [14]. These tools use linguistic similarity matches between concepts for initiating the merging or alignment process and then use the underlying ontological structures in 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 and colleagues for the Ontolingua ontology editor: Chimaera [28]. 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 [25] and Doan and colleagues [8] 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 [5] translation system for symbolic knowledge, Kiryakov and colleagues' OntoMap portal [24] for mapping linguistic ontologies, the OBSERVER system [29] by Mena and colleagues for information integration, Gangemi and colleagues' [15] ONIONS methodology for medical ontologies, Visser and Tamma's heterogeneity categorisation [36], and the reports from Pinto and colleagues [31] and Fridman Noy and Hafner in [12].

7 Summary

In this paper we presented a novel method and a theory for ontology mapping. We formalised the notion of ontology, ontology morphism, 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 in large-scale ontology mapping efforts and are continue researching fruitful extensions of it, such as, ontology merging, reasoning about ontology evolution, and inclusion of ontological axioms.

Acknowledgements. 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.

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 Tracts in Theoretical Computer Science 44. Cambridge University Press, 1997.
4. T. Berners-Lee, J. Hendler., and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
5. H. Chalupksy. OntoMorph: A Translation System for Symbolic Knowledge. In *Proceedings of the 17th International Conference on Knowledge Representation and Reasoning (KR-2000), Colorado, USA*, April 2000.
6. 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.
7. K. Devlin. *Logic and Information*. Cambridge University Press, 1991.
8. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *Proceedings of the 11th International World Wide Web Conference (WWW 2002), Hawaii, USA*, May 2002.
9. J. Domingue. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. In *Proceedings of the 11th Knowledge Acquisition, Modelling and Management Workshop, KAW'98, Banff, Canada*, April 1998.
10. F. Dretske. *Logic and the Flow of Information*. MIT Press, 1981.
11. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–728, June 1997.
12. N. Fridman Noy and C.D. Hafner. The State of the Art in Ontology Design: A Survey and Comparative Review. *AI Magazine*, 18(3):53–74, 1997.
13. N. Fridman Noy and M. Musen. SMART: Automated Support for Ontology Merging and Alignment. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modelling and Management (KAW'99), Banff, Canada*, October 1999.
14. N. Fridman Noy and M. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence, (AAAI'00), Austin, TX, USA*, July 2000.
15. A. Gangemi, D. Pisanelli, and G. Steve. Ontology Integration: Experiences with Medical Terminologies. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontology in Information Systems, FOIS'98, Trento, Italy*, pages 163–178, June 1998.
16. B. Ganter and R. Wille. *Formal Concept Analysis: mathematical foundations*. Springer, 1999.
17. R. Genesereth and R. Fikes. *Knowledge Interchange Format*. Computer Science Dept., Stanford University, 3.0 edition, 1992. Technical Report, Logic-92-1.
18. W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, and M. Musen. Knowledge Modelling at the Millennium (The Design and Evolution of Protege-2000). In *proceedings of the 12th Knowledge Acquisition, Modelling, and Management(KAW'99), Banff, Canada*, October 1999.

19. V. Gupta. *Chu Spaces: A Model of Concurrency*. PhD thesis, Stanford University, 1994.
20. J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. Encapsulation and Composition of Ontologies. In *Proceedings of the AAAI'98 Workshop on Information Integration, Madison, WI, USA*, July 1998.
21. Y. Kalfoglou. *Deploying Ontologies in Software Design*. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, June 2000.
22. R. Kent. The Information Flow Foundation for Conceptual Knowledge Organization. In *Proceedings of the 6th International Conference of the International Society for Knowledge Organization (ISKO), Toronto, Canada*, August 2000.
23. R. Kent. The IFF Foundation Ontology, v.1.0. In *Proceedings of the IJCAI'01 Workshop on the IEEE Standard Upper Ontology, Seattle, WA, USA*, August 2001.
24. A. Kiryakov, K. Simov, and M. Dimitrov. OntoMap: Portal for Upper-Level Ontologies. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS'01), Ogunquit, Maine, USA*, October 2001.
25. M. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proceedings of the 14th International FLAIRS conference, Key West, FL, USA*, May 2001.
26. O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C recommendation, W3C, February 1999.
27. B. McBride. Jena: Implementing the RDF model and syntax specification. Technical report, HP Labs at Bristol, UK, December 2001. Semantic Web Activity, http://www.hpl.hp.com/semweb/.
28. D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Colorado, USA*, April 2000.
29. 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, *Proceedings of the 1st International Conference on Formal Ontology in Information Systems (FOIS'98), Trento, Italy*, pages 269–283. IOS Press, June 1998.
30. E. Motta. *Reusable Components for Knowledge Models: Case Studies in Parametric Design Problem Solving*, volume 53 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1999.
31. S. Pinto, A. Gómez-Pérez, and J. Martins. Some Issues on Ontology Integration. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden*, August 1999.
32. V. R. Pratt. The Stone gamut: A coordination of mathematics. In *Logic in Computer Science*, pages 444–454. IEEE Computer Society Press, 1995.
33. I. Schmitt and G. Saake. Merging Inheritance Hierarchies for Database Integration. In *Proceedings of the 3rd International Conference on Cooperative Information Systems (CoopIS'98), New York, USA*, August 1998.
34. M. Schorlemmer. Duality in Knowledge Sharing. In *Proceedings of the 7th International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida,USA*, January 2002.
35. G. Stumme and A. Maedche. Ontology Merging for Federated Ontologies on the Semantic Web. In *Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001), Viterbo, Italy*, September 2001.
36. P. Visser and V. Tamma. An experiment with ontology-based agent clustering. In *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods, Stockholm, Sweden*, August 1999.