"What's **not** in a name?" Initial Explorations of a Structural Approach to Integrating Large Concept Knowledge-Bases

Alex Borgida
Dept. of Computer Science
Rutgers University

Ralf Küsters
Theoretical Computer Science
RWTH Aachen

August 10, 1999

Technical Report DCS-TR-391

Abstract

We are interested in providing semi-automatic support for the task of integrating large knowledge bases (KBs) through the use of structural information in description logics. For that reason we set up a formal framework for the integration of KBs which enables us to investigate the potential and limits of using structural information (in contrast to just thesaurus information involving identifier strings).

Structurally similar concept descriptions (role chains) are related by integration mappings, which must satisfy a previously proposed notion of "conflict-freeness". It turns out that computing integration mappings is closely related to unification and matching of concept descriptions (and is intractable even for simple description logics).

As a formal basis for the limitations of structural information we introduce the notion of indistinguishable concepts. We state the connection between indistinguishable concepts and integration mappings theoretically and also evaluate the problem empirically, by looking for indistinguishable concepts in a sizeable ontology (GALEN), and integration mappings from GALEN to itself. An interesting discovery is that roughly two thirds of the concepts are distinguishable by their structural properties, indicating that semantic information is indeed being encoded in the knowledge representation, not just in the name.

1 Introduction

Given two ontologies/terminologies — collections of terms and their "meanings" as used in some universe of discourse (UofD), our general task is to *integrate* them into a single ontology, which captures the meanings of the original terms and their inter-relationships.

This problem is motivated by several application scenarios:

- First, such ontologies have been and are being developed independently by multiple groups for knowledge-based and other applications. Among others, medicine is an area in which such ontologies already abound [RZStGC, CCHJ94, SCC97].

- Second, a traditional step in database design has been so-called "view integration": taking the descriptions of the database needs of different parts of an organization (called "external views"), and coming up with a unified central schema (called the "logical schema") for the database [BLN86]. Although the database views might be expressed in some low-level formalism, such as the relational data model, one can express the semantics (meta-data) in a more expressive notation, which can be thought of as an ontology. Then, the integration of the ontologies can guide the integration of the views.

- Finally, databases and semistructured data on the internet provide many examples where there are multiple, existing *heterogeneous information sources*, for which uniform access is desired. To achieve this goal, it is necessary to relate the contents of the various information sources. The approach of choice has been the development of a single, integrated ontology, starting from separate ontologies capturing the semantics of the heterogeneous sources[Kas97, CDGL$^+$98].

Of course, we could just take the union of the two ontologies, and return the result as the integration. However, except for the case when the ontologies had absolutely nothing to do with each other, this seems inappropriate. Therefore part of our task will to be explore what it means to "integrate" two ontologies. To help in this, we will in fact assume here that *the ontologies are describing exactly the same aspects of the universe of discourse* (UofD), leaving for a separate paper the issue of dealing with partially overlapping ontologies.

So why wouldn't the two ontologies be identical when modeling the same UofD? The answer is that there are multiple ways of modeling the same situation. For example

- choice of identifiers: identifiers for the same UofD term may be simple morphological variants (e.g. PATIENT vs. PATIENTS), synonymous (e.g., DOCTOR vs. PHYSICIAN), or related, but not synonymous terms (OFFICE-VISIT vs. APPOINTMENT).

- choice of modeling detail: (i) an address might be modeled as a single string or, in greater detail, as an object with attributes street, city, state etc.; (ii) the office phone number of an employee may be modeled directly as an attribute office-phone, or the employee might have an associated office (attribute hasOffice), which in turn has its own phone value.

- reification: the relationship between a patient and her doctor might be modeled by the role treatedBy, or there might be a new concept, TREAT, with functional attributes who (a doctor) and whom (a patient).

Therefore, a central part of ontology integration is obtaining some (maximal?) collection of assertions that relate the terms in them. (Another part is eliminating redundancies.) A simple form of such assertions is a mapping from single identifiers in one ontology to expressions in the

other ontology which correspond to the same notion — an integration mapping. (If the ontologies are not modeling exactly the same UofD because different levels of abstraction are allowed, then the integration mapping would indicate additional relationships, like hypernomy.)

The above problems and ideas are not new, as we shall see in the next section. One distinguishing aspects of the present work is that the terminology will be assumed to consist of concept descriptions in a *description logic* (DL) [Bor95] – a formalism that allows semantic relationships to be captured using complex *nested* structures, definitions and attribute hierarchies —unlike previously used formalisms such as the relational, ER, and OO data models. A second differentiating feature is the assumption that the ontologies are *large* (containing on the order of thousands of terms), and possibly in different languages.

This means that we cannot ask the user to generate interschema assertions on their own, because the KBs are too large. Therefore our eventual goal is to provide tools which ask users to verify certain highly plausible inter-schema relationships. Research on such tools has, in the past, relied heavily on the matching of identifiers — a not unreasonable heuristic. However, we want to first understand to what extent ontologies explicate the meaning of their terms independent of the choice of identifiers. This is important because to the extent that the meaning of terms is captured purely *formally*, an integration mapping might be found automatically.

The present paper therefore looks at formal ways of characterizing ontology integration, specifically integration mappings, and then considers algorithmic aspects of finding such mappings.

To study the problem empirically, we also carry out experiments with a real ontology – GALEN [RZStGC] – containing close to 3000 concepts. In fact, we will set up an idealized integration experiment, where the ontology will be integrated with a copy of itself (in which the identifiers have been renamed).

To summarize, this paper can therefore be considered as an investigation into the limits of automatic ontology integration under idealized circumstances. However, as mentioned earlier, our long term goal is the development of tools for supporting humans in large-scale ontology integration under de-idealized circumstances.

2 Related work

The problem of taking two or more pre-existing world-descriptions and merging them into a single, globally unified one has been considered in several disciplines within computer science. We will review here mostly work that has appeared in the database literature (see [BLN86] for an overview of early work in this field).

We mention from the start that we will be ignoring one issue that has received considerable attention lately: namely whether the integration should be carried out completely at first, generating a new schema; or whether, due to continuous changes in the original schemas, it is better to follow a more "loosely-coupled" [Goh96, Kas97, CDGL$^+$98] strategy, where the system maintains inter-ontology links (which can be dynamically changed or discovered), and then uses these during the access process.

2.1 Database schema integration

Since the early days of the ANSI database design methodology, developers were supposed to obtain from different stake-holders *external views*, representing each group's perspective and interest in the company's UofD. Then, the database designer was supposed to reconcile these into a single

global schema, which supports the activities of the entire enterprise. Clearly, it is essential that all stake-holders be able to re-derive their own view from the global, integrated schema.

More recently, the development of heterogeneous and federated databases requires the design of a conceptual schema that can be presented to the user for the purposes of querying, and which hides the essential independence of the component databases.

The traditional integration methodology, as described in [BLN86] involves 4 steps:

1. Choice of integration strategy.
   In the case when more than two schemas are to be merged, some strategy needs to be chosen for merging them. Choices include various orderings of binary merges, as well as "one-shot" merges of entire sets of schemas at a time. The binary strategies motivate the potential need for a non-symmetric *"directional"* integration operator, since one of the schemas seems more important (if it was already the result of lots of other integrations).

   In this paper, we will consider only the integration of two ontologies.

2. Schema comparison.
   This phase requires determining *correspondences* between the elements of the two schemas. When the correspondences are not the identity relationship, they are traditionally called "conflicts" (a misnomer, since conflict should probably be reserved for irreconcilable differences – logical inconsistencies). Among the conflicts, it is traditional to distinguish

   - Name conflicts, such as homonyms (same identifier denotes distinct UofD notions), and synonyms (different identifiers denoting identical UofD notions).
   - Structural conflicts, arising because of different ways of modeling the same aspect of the UofD. The kinds of such conflicts depend on the conceptual modeling language, and considerable work has been devoted to finding exhaustive catalogs of such conflicts for the relational data model (augmented by various dependencies) and the Entity-Relationship model (or its extensions).

3. Schema conforming.
   One attempts to resolve the previously discovered conflicts by finding an *"equivalent"* representation for one (or both) of the conflicting schema elements. There are a variety of notions of equivalence that have been mentioned in the literature:

   - mapping: there is a one-one correspondence between the instances of the two schemas;
   - behavioural: for every instance of the first representation there is a corresponding instance of the second, which gives the same answer to any query;
   - transformational: one schema can be obtained from the other through a sequence of primitive transformations, each of which is, by definition, equivalence preserving.

4. Schema merging and restructuring.
   After conforming, common concepts are superimposed. In the resulting mixed schema one than looks for additional relationships between concepts, other than equivalence; these include various set-based relationships such as containment, compatibility and disjointness of concepts.

   The superimposed schema is then subject to additional transformations in order to capture these interschema assertions and to eliminate redundancies. In non-relational data models, this phase normally results in the introduction of new concepts in the IsA or aggregation hierarchy, which capture the various interschema assertions discovered at the beginning.

The survey of Batini et al [BLN86] (which covered the period up to 1984 or so) was brought up-to-date to 1992 by Francalanci and Pernici [FP93].

Due to lack of time and space, this report also does not review the relevant recent literature on tool support for database integration, nor the work in artificial intelligence, especially natural language processing, dealing with ontology merging.

Instead, we will concentrate on special issues that are of interest to us, including different notions of schema and concept equivalence.

2.2 What is a good integration?

The desired intuitive properties of the integrated schema seem to include

- ability to recover *all* the information stored under the original schemas;
- "correctness" of the resulting schema: it must satisfy the inherent constraints of the data model;
- minimality: there should be no redundant elements in the schema; a special case of this would be the absence of schema elements that can have no instances, because its specification is equivalent to the concept $false$.
- the ability to propagate updates from the original schemas to the integrated schema;
- possibly the converse ability to propagate updates from the integrated schema to the (appropriate) components;
- naturalness/understandability: a qualitative preference criterion for humans;

In order to explain the various approaches (more or less formal) in the literature, let us assume in the following that every view schema V, consists of a set of predicates P_V, and a set of logical constraints C_V describing the semantics of these predicates. (Hence, we shall occasionally treat a schema as a logical theory, with a set of models.) The set of constraints is expressed in a language dictated by the data model; (e.g., for the relational model, they might be key constraints and dependencies of some fixed kind).

Integration of information sources is based on the general notion of *information capacity* introduced by Hull [Hul86]:

Definition 1 The schema W *is dominated by* V if there is a total 1-1 function f from the models of W to those of V.

Schema W and V have *equivalent information capacity*, if they dominate each other.

This definition turns out to be an insufficient basis for expressing the intuitive relationship between a view and the global schema that dominates it since f can perform too much computation. Therefore, in general one looks for limitations on f and uses dominance as an auxiliary tool for integration.

Integration using assertions The most general and appealing form of the view integration problem is presented by Biskup [BC86], who characterize view integration as a process with

- input: two schemas, V1 and V2, together with a set $\mathcal{A}$ of *integration assertions (IAs)*: these assertions essentially relate the terms in the two views by enforcing complex relationships between the predicates in the two views. IA are expressed in a second language, the integration language. (Assumption: identifiers in V1 and V2 are distinct, to avoid homonym problems.)

- intermediate result: $Comb$, equal to $V1 \cup V2 \cup \mathcal{A}$
- output: a schema G, and functions q, q' (expressed in a third language — the derivation/query language), such that $q(models(G)) = models(Comb)$ and $models(G) = q'(models(Comb))$; i.e., each view state/model can be obtained from the state of the global schema, and (conversely), all states of the global schema correspond to combinations of the states of the views.

In this case, presumably the minimality condition is folded into the notion of schema, where there should be no redundancy in the semantic specification C_G.

Therefore the general task of schema integration has the following steps:

a) gather the interschema assertions;

b) transform $Comb$, eliminating IAs which are not in the language of schema definitions, while preserving the set of models;

c) minimize the resulting schema

d) specify the query q for deriving the views from the global schema.

The above scheme can then be investigated for a variety of languages for specifying schemas, IAs, and queries. We summarize here some related research based on the IA language used.

General: Catarci and Lenzerini [CL93] use DLs (see Section 3 for an introduction) for describing the schema, and also for IAs, allowing arbitrary containment assertions of the form $D1 \sqsubseteq D2$ for this purpose. If schemas can also contain arbitrary containments of the above form, then there is nothing more to do after step (a), except the standard classification of concepts, which automatically detects synonyms, including synonyms of the incoherent concept (which are of course undesirable). More recently, Calvanese et al. [CGL$^+$98] have used a much more expressive description logic for information integration, one which supports n-ary relations and other features desirable for practical applications.

Set theoretic containment: Biskup and Convent [BC86] consider the integration of relational databases, and use 4 kinds of IAs for pairs of relations in different views; these deal with identity (possibly using the selection operation), disjointness and containment. For example, the doctors in one view (identified by their name and phone), correspond to the New Jersey health providers in the other view (identified by their IdNr)
DOCTOR[name,phone] $\equiv$ Select$_{state='nj'}$(HEALTH-PROVIDER[IdNr])
Their paper discusses how each of these constraints can be "discharged" by modifying the schema $Comb$, and some circumstances under which this cannot be done.

Kashyap et al [MKSI96, Kas97] describe their information sources using DLs again (a relatively weak one in this case), and utilize 3 kinds of IAs between *identifiers* in two ontologies: synonym, hyponym and hypernym, indicating equality or containment of intensions.

Identifier mapping: Johannesson and colleagues [EJ95] restrict IAs to have the form <identifier in V1>=<formula in V2>. This restricted form (which has synonyms as a special case) has the advantage of making the elimination of IAs relatively easy, since all we need to do is substitute formulas for names in C_2 (and then eliminate redundant constraints).

To summarize, it appears that the form of the IAs allowed depends a lot on the technology available for eliminating them from $Comb$ (hence on the schema language).

Transformational approaches to integration One approach to integration is to union the schemas, and then apply restructuring/merging transformations, which are designed to identify common structures that can be combined in order to simplify the result. For example, the shared attributes of two entities are factored out into a superclass [DH84].

A different approach is to superimpose/merge constructs judged to be equivalent [LNE89], and to transform structurally inequivalent representations of the same concept into mergeable forms [GLN92]. Much of this work has the flavor of considering various syntactic combinations of schema modeling constructs, and finding conditions under which one can be used to represent the same real-world notion as the other.

The theoretical foundation for the work on schema transformations is provided by Miller et al [MIR93], who examine a large variety of such transformations and show that they (should) obey the rule of information capacity preservation (see Definition 1).

Unfortunately, little is said anywhere about the formal theory of schema *integration* using transformations – what happens after identical schema fragments are superimposed.

3 Concept Description Languages/Logics

DLs have an object-centered ontology. Individual DL objects (which have immutable identity) may be related to other objects via *roles*, and may be grouped into *concepts*. For example, C45 (an instance of concept CAR) can be related to its maker, Gm (an instance of MANUF and US-CORP) by having Gm be a filler of the madeBy attribute. By default, an individual object may be related to zero or more other objects by an role – e.g., ownedBy for C45 may include several entities, indicating multiple ownership.

DLs start from atomic/primitive concept and role names, and combine these into composite concepts using *concept constructors*. Some concept constructors are the familiar boolean connectives: CAR⊓ BLACK-OBJECT denotes the set of cars that are also black objects. Other concept constructors are more specially suited to represent conceptual models of application domains. For example, one can express cardinality constraints on roles using constructors **at-least** and **at-most**: ≥ 2 ownedBy denotes objects with at least two owners. The values of roles can be restricted in various ways, including by universal quantification, as in ∀ownedBy.PERSON which denotes objects owned only by persons.

A useful property of DLs is the ability to nest composite descriptions, so that

$$\forall \mathsf{madeBy}.(\mathsf{MANUF} \sqcap \forall \mathsf{postedProfit}.\mathsf{HIGH})$$

describes objects made by MANUFs who posted a HIGH profit.

Table 1 contains a subset of DL constructors (see, e.g., [Bor95] for other constructors). The DL called $\mathcal{FL}_0$ allows for the first four constructors in the table; $\mathcal{FLE}$ extends $\mathcal{FL}_0$ by existential restrictions. In the sequel, we occasionally use $\forall r_1 \cdots r_n.C$ as abbreviation for $\forall r_1.\forall r_2. \cdots \forall r_n.C$.

Formally, the semantics of a concept description is defined in terms of an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$. The domain Δ of $\mathcal{I}$ is a non-empty set of individuals and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name P to $P^I \subseteq \Delta$ and each role name r to a binary relation $r^I \subseteq \Delta \times \Delta$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column of Table 1.

DLs can not only express information about a domain of discourse, but are also logics that provide reasoning services. A basic operation is determining if some concept C *is subsumed by* another, D ($C \sqsubseteq D$). For example, CAR ⊓ ∀ownedBy.PERSON ⊑ CAR. Formally, subsumption is defined as follows:

Construct name	Syntax	Semantics
concept name P	P	$P^I \subseteq \Delta$
top-concept $\top$	$\top$	Δ
conjunction	$C \sqcap D$	$C^I \cap D^I$
value restrictions	$\forall r.C$	$\{x \in \Delta \mid \forall y : (x,y) \in r^I \rightarrow y \in C^I\}$
existential restrictions	$\exists r.C$	$\{x \in \Delta \mid \exists y : (x,y) \in r^I \wedge y \in C^I\}$
primitive negation	$\neg P$	$\Delta \setminus P^I$
incoherent concept	$\bot$	$\emptyset$
atmost restrictions	$(\leq n\, r)$	$\{x \in \Delta \mid \#\{y \mid (x,y) \in r^I\} \leq n\}$
atleast restrictions	$(\geq n\, r)$	$\{x \in \Delta \mid \#\{y \mid (x,y) \in r^I\} \geq n\}$

Table 1: Syntax and semantics of concept descriptions

Definition 2 Let C, D be concept descriptions.
D *subsumes* C (for short $C \sqsubseteq D$) iff $C^I \subseteq D^I$ for all interpretations $\mathcal{I}$.
D is *equivalent* to C (for short $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., $C^I = D^I$ for all interpretations $\mathcal{I}$.

Often, the question of subsumption is asked in the context of so-called schemas which state necessary (and possibly sufficient) conditions for concept names. A *schema* S consists of a finite set of axioms of the form $A \sqsubseteq C$ and $A \equiv C$ where A is a concept name and C is a concept description; S is called $\mathcal{FL}_0$-schema if the concept descriptions C are $\mathcal{FL}_0$-concept descriptions. Analogously, we define $\mathcal{FLE}$-schemas. A *model* $\mathcal{I}$ of a schema S is an interpretation such that all axioms are satisfied, i.e., for all axioms $A \sqsubseteq C$ ($A \equiv C$) in S, $\mathcal{I}$ satisfies $A^I \subseteq C^I$ ($A^I = C^I$). Now, C is *subsumed by* D *w.r.t. a schema* S ($S \models C \sqsubseteq D$ for short) iff $C^I \subseteq D^I$ for all models $\mathcal{I}$ of S.

The various DLs investigated so far have different kinds of implementations: the most widely used, including CLASSIC[ABM+89] and LOOM[Mac87], are implemented in a "normalize then compare" approach, which is quite different than standard theorem proving, and relies on finding a normal form for descriptions that detects nested incoherences, explicates implicit concepts, and removes redundancies. A second family of DLs, typified by CRACK [BFT95] and IFACT [HPSar], is implemented with tableaux-like refutation theorem proving techniques, albeit ones specially made for DLs. Recently, several powerful decidable DLs [CGL99] have been identified, which are closely related to Propositional Dynamic Logic; therefore theorem provers of the later could be used for reasoning, although the prefered trend is to extend tableaux techniques to handle them.

It is important to note that a key focus of reasoning research, especially for DLs, has been the *trade-off* between expressiveness of a language and the cost of reasoning with it. Specifically, there are detailed results about the *decidability and computational complexity* of reasoning with various subsets of constructors (and restrictions of them). For example, CLASSIC [ABM+89] has a complete subsumption algorithm that is $O(n^3)$. Also, recent empirical evidence [HPSar] suggests that special optimizations make tableaux techniques work quite fast on "normal" knowledge bases, though the problem is worst-case exponential.

4 A Formal Framework for Integration

Integration mappings, which induce interschema assertions, form the core of our framework. The goal is to compute such mappings, and then offer them to the human user as potential interschema

assertions. Integration mappings relate sets of elements in one schema to sets of descriptions in the other schema. This enables us to take into account more structural information, encoded in the schemas, than if we were considering only the structural similarity between two elements at a time.

Formally, we express interschema assertions between two schemas S_1 and S_2 over disjoint signatures Σ_1 and Σ_2, respectively. The interschema assertions we are dealing with generally connect two concepts (roles) C and D, over signatures $\Sigma 1$ and $\Sigma 2$ respectively. As we saw in Section 2, traditionally the relationship between C and D is usually in the set {synononymy, hypernomy, hyponymy}, which can be captured by DL assertions of the form $C \sqsubseteq D$ and/or $D \sqsubseteq C$. Antonymy relations can be captured by assertions of the form $C \sqcap D \equiv \bot$ (or $C \sqsubseteq \neg D$, if the language supports negation/complement).

As we saw earlier, the over-riding intuition is that information in the original schemas should not be lost when they are integrated. This means that interschema assertions specified by the user do not reduce the information capacity of the schemas to be integrated; in other words, the set of models of the views/schemas shall not be reduced by the IAs. We will formalize this by using the notion of conflictfreeness that has been introduced in [BC86] for the relational data model, and also used in [EJ95] for a FOL data model.

In order to give a formal definition of conflictfreeness, we need the set of all models $I(S)$ of a schema S. Furthermore, if Σ' denotes a subset of the signature Σ of S then we refer to $I(S)[\Sigma']$ as the projection of the models in $I(S)$ to Σ'.

Definition 3 Let $\mathcal{A}$ be a set of interschema assertions between S_1 and S_2. Then S_1 and S_2 are *conflictfree* w.r.t. $\mathcal{A}$ if and only if $I(S_1 \cup S_2 \cup \mathcal{A})[\Sigma_i] = I(S_i)$ for $i = 1, 2$.

Since $I(S_1 \cup S_2 \cup \mathcal{A})[\Sigma_i] \subseteq I(S_i)$ is always true, the interesting inclusion is in the other direction, which says that every model $\mathcal{I}$ of S_i can be extended to a model $\mathcal{I}'$ of $S_1 \cup S_2 \cup \mathcal{A}$ where *extension* means that $\mathcal{I}'$ and $\mathcal{I}$ coincide on the signature of S_i, say S_1, but $\mathcal{I}'$ might have a bigger domain and in addition interprets the identifiers of S_2. Intuitively, the inclusion means that the assertions in $\mathcal{A}$ actually describe a certain connection between S_1 and S_2 as opposed to imposing additional constraints on the schemas. If they restricted the models of S_1 or S_2 then these assertions could not just be a description of the *connections* between these two schemas: they would be adding additional knowledge too! So conflictfreeness can be seen as a necessary condition for interschema assertions to describe relationships between expressions in S_1 and S_2. Additional knowledge is added, for example, when relating not necessarily disjoint concepts in S_1 to disjoint concepts in S_2. The underlying assumption of our approach is that the schemas contain the *complete* structural information expressible in the language of choice . Thus, if concepts in S_1 are not defined to be disjoint, it would not make sense to relate them to disjoint concepts in S_2.

In the following, we shall concentrate on a subset of IAs, which try to give the meaning of an identifier in one of the schemas in terms of descriptions in the other schema. Formally, these assertions have the form $N \equiv C$, where N is a concept name (role name) in Σ_1, C is a concept description (role chain) over Σ_2. (Symmetrically, we could also give the meaning of identifiers B, from Σ_2, in terms of $S1$.) The collection of such assertions can be represented by a partial mapping σ from Σ_1 into T_{Σ_2}, where $\sigma(N)$ is C. Such a mapping σ is called an *integration mapping* from S_1 into S_2. Conversely, such a mapping induces a set $\mathcal{A}_\sigma$ of interschema assertions

$$\mathcal{A}_\sigma := \{N \equiv \sigma(N) \mid N \in \mathsf{domain}(\sigma)\}.$$

The meaning of a term used lies in the intuition of the user. In the schema, some portion of the meaning is captured by the name and some by the structure of the concept. The structure of

a concept includes its sub- and super-concepts, the various role restrictions, and its use in other concept specifications.

We suppose that intuitively identical concepts (roles) in different schemas have similar structures; hence σ should map an identifier on a structurally similar concept (role expression). This similarity is captured by the notion of conflictfreeness, introduced above.

Definition 4 A partial mapping σ from Σ_1 into T_{Σ_2} is called an *integration mapping* from S_1 into S_2 if and only if it is conflictfree, i.e., S_1 and S_2 are conflictfree w.r.t. $\mathcal{A}_\sigma$.

In the following, we will discuss the notion of conflictfreeness in order to give a feel to what extent this notion rules out certain unintuitive integration mappings. To this end, we will start with a very simple example.

Example 5 Let the schemas S_1 and S_2 be defined as follows:

S_1: S_2:

$$\begin{aligned} \mathsf{HeartClinic} &\sqsubseteq \mathsf{Clinic} \\ \mathsf{PsychiatricClinic} &\sqsubseteq \mathsf{Clinic} \end{aligned} \qquad \begin{aligned} \mathsf{Heart} &\sqsubseteq \mathsf{Hospital} \\ \mathsf{Psychiatric} &\sqsubseteq \mathsf{Hospital} \end{aligned}$$

Through this example, we can illustrate that an integration mapping must be i) injective, which, roughly speaking, means that two different concepts in S_1 may not be mapped on the same concept in S_2 (see below for a formal definition) and ii) must preserve the subsumption relationships.

Concerning injectiveness, if σ mapped HeartClinic and Clinic on the same concept in S_2, say Heart, then such a σ would not be conflictfree: One could easily come up with a model of S_1 where HeartClinic and Clinic do not have the same interpretation. In this case, it is not possible to extend that model to a model of S_2 together with the two assertions induced by σ.

In order to see that σ must preserve subsumption relationships, consider a function σ mapping Clinic on Heart and HeartClinic on Hospital. Then, any model of S_1 where the interpretation of HeartClinic is a proper subset of the interpretation for Clinic can not be extended to a model of S_2 and the assertions $\mathcal{A}_\sigma$ for σ.

These two observation are generalized in the following two lemmas. First, we need to define injectivity. A mapping σ from S_1 into S_2 is called *injective* if and only if for all identifiers A, B in the domain of σ, $S_2 \models \sigma(A) \equiv \sigma(B)$ implies $S_1 \models A \equiv B$.

Lemma 6 Let σ be an integration mapping from S_1 into S_2. Then σ is injective.

PROOF. Assume that σ is not injective. In this case, there are identifiers A, B such that $S_2 \models \sigma(A) \equiv \sigma(B)$ but $S_1 \models A \not\equiv B$. Thus, there exists a model $\mathcal{I}$ of S_1 such that $A^I \neq B^I$. If $\mathcal{I}'$ is an extension of $\mathcal{I}$ with $\mathcal{I}' \models S_1 \cup S_2 \cup \mathcal{A}_\sigma$, then since $A \equiv \sigma(A)$ and $B \equiv \sigma(B)$ are assertions in $\mathcal{A}_\sigma$ we know $\sigma(A)^{\mathcal{I}'} \neq \sigma(B)^{\mathcal{I}'}$, which is a contradiction to the fact that $\mathcal{I}'$ is a model of S_2 and $S_2 \models \sigma(A) \equiv \sigma(B)$. ∎

The next lemma shows that integration mappings must preserve the subsumption relationships.

Lemma 7 Let σ be an integration mapping from S_1 into S_2. Then, for all $C, D \in T_{\mathsf{domain}(\sigma)}$, $S_1 \models C \sqsubseteq D$ is equivalent to $S_2 \cup \mathcal{A}_\sigma \models C \sqsubseteq D$ (which is equivalent to $S_2 \models \sigma(C) \sqsubseteq \sigma(D)$).

PROOF. Assume, there exist $C, D \in T_{\mathsf{domain}(\sigma)}$ with $S_1 \models C \sqsubseteq D$ but $S_2 \cup \mathcal{A}_\sigma \not\models C \sqsubseteq D$. Then there exists $\mathcal{I} \models S_2 \cup \mathcal{A}_\sigma$ with $C^I \not\subseteq D^I$. This means that $\mathcal{I}$ cannot be extended to a model $\mathcal{I}'$ of S_1 since otherwise we would have $C^I = C^{I'} \subseteq D^{I'} = D^I$ in contradiction to the assumption.

Conversely, assume that there are $C, D \in T_{\mathsf{domain}(\sigma)}$ with $S_2 \cup \mathcal{A}_\sigma \models C \sqsubseteq D$ but $S_1 \not\models C \sqsubseteq D$. Then, there exists a model $\mathcal{I}$ of S_1 with $C^I \not\subseteq D^I$. Analogously to the only-if direction, one can show that $\mathcal{I}$ cannot be extended to a model of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$. ∎

These lemmas show that conflictfreeness rules out some unintuitive mappings. Nevertheless, due to the restricted structural information provided by schemas, it is not possible to rule out all undesirable mappings. In Section 5, we consider some fundamental limitations of structural information, which will be confirmed by experimental results. As a first example, consider Example 5. The mapping

$$\begin{aligned} \mathsf{Clinic} &\mapsto \mathsf{Hospital} \\ \mathsf{HeartClinic} &\mapsto \mathsf{Heart} \\ \mathsf{PsychiatricClinic} &\mapsto \mathsf{Psychiatric} \end{aligned}$$

cannot be distinguished from the one that switches the images for HeartClinic and PsychiatricClinic. Both mappings are conflictfree, but only the former is the intended one.

In the remainder of this section we shall take a look at weaker and stronger constraints on integration mappings.

Weaker constraints on integration mappings

In [EJ95], a weaker version of conflictfreeness has been considered, which we will call S_2-conflictfreeness.

Definition 8 A partial mapping σ from Σ_1 into T_{Σ_2} is called *S_2-conflictfree* if and only if $I(S_1 \cup S_2 \cup \mathcal{A}_\sigma)[\Sigma_2] = I(S_2)$.

Thus, one only requires that S_2-models can be extend to models of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$ as opposed to both S_1- and S_2-models. As a result, the statement of Lemma 7 has to be restricted as follows.

Lemma 9 Let σ be an S_2-conflictfree mapping from S_1 into S_2. Then, for all $C, D \in T_{\mathsf{domain}(\sigma)}$, $S_1 \models C \sqsubseteq D$ implies $S_2 \cup \mathcal{A}_\sigma \models C \sqsubseteq D$ (which is equivalent to $S_2 \models \sigma(C) \sqsubseteq \sigma(D)$).

In the sequel, we will illustrate that S_2-conflictfreeness is a strictly weaker condition than (general) conflictfreeness. The first observation is that mappings from S_1 into S_2 no longer have to be injective. Referring to Example 5, when mapping all concepts of S_1 onto Hospital in S_2 one obtains a non-injective S_2-conflictfree mapping.

More generally, one can construct mappings from S_1 into S_2 that intuitively are not related to S_2 whatsoever: For this, suppose that S_1 is an acyclic $\mathcal{FL}_0$-schema with axioms of the form $A \sqsubseteq C$. Furthermore, w.l.o.g. assume that for every concept name A in S_1 there exists at most one axiom $A \sqsubseteq N_A$ in S_1. Now, we iteratively expand S_1 until the right hand-side of the axioms only contain concept names for which there are no axioms that have this concept as left hand-side. Expanding means that a concept name A occurring somewhere on the right hand-side of an axiom is replaced by its necessary condition N_A in case such an axiom $A \sqsubseteq N_A$ exists for A. Let S_1' denote the expanded version of S_1 with axioms of the form $A \sqsubseteq N_A'$. Finally, let σ be a mapping from S_1 into S_2 where all concept names that do not occur on the left hand-side of some axiom in S_1 are mapped on some concept description over Σ_2 and all role names are mapped on some

role chain over Σ_2. Furthermore, we define $\sigma(A) := \sigma(N'_A)$. Then it is easy to verify that σ is an S_2-conflictfree mapping. This mapping could be constructed regardless of S_2, and therefore, is not a useful integration mapping. Note, that in general such a σ is not conflictfree. An example shall illustrate the construction.

Example 10 Let S_1 consist of the already expanded axiom

$$\mathsf{HeartClinic} \quad \sqsubseteq \quad \forall \mathsf{patients.HeartDisease.}$$

and let S_2 be

$$\mathsf{Heart} \quad \sqsubseteq \quad \forall \mathsf{patients\text{-}have.HeartProblems.}$$

The mapping σ defined by

$$\begin{aligned} \mathsf{patients} &\mapsto \mathsf{patients\text{-}have} \\ \mathsf{HeartDisease} &\mapsto \mathsf{Heart} \\ \mathsf{HeartClinic} &\mapsto \forall \mathsf{patients\text{-}have.Heart} \end{aligned}$$

is a S_2-conflictfree mapping. However, σ is not conflictfree since one can construct a model of S_1 such that the interpretation of HeartClinic is a proper subset of the interpretation of its necessary conditions. Such an interpretation cannot be extended to a model of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$ as such a model would require the interpretation of HeartClinic and its necessary conditions to be equal.

These examples show that S_2-conflictfreeness is a condition too weak to rule out unintended integration mappings.

Stronger constraints on integration mappings

A natural question to ask is whether conflictfreeness is sufficient to rule out as many unintuitive mappings as possible. It turns out that the answer is no. Referring to Example 10, σ defined by

$$\begin{aligned} \mathsf{HeartClinic} &\mapsto \forall \mathsf{patients\text{-}have.Heart} \\ \mathsf{HeartDisease} &\mapsto \forall \mathsf{patients\text{-}have.HeartProblems} \\ \mathsf{patients} &\mapsto \mathsf{patients\text{-}have} \end{aligned}$$

is a conflictfree mapping from S_1 into S_2, although, it is not an intuitive one.

One way of solving these kinds of problems is to impose some kind of "Occam's razor" preference for simpler explanations, which would lead us to prefer "small" mappings.

Another effective way of avoiding unintended mappings is to include additional interschema assertions specified by the user, and thus provide additional semantics that eliminate unreasonable mappings.

Definition 11 An integration mapping σ from S_1 into S_2 w.r.t. interschema assertions $\mathcal{A}$ has to satisfy two conditions:

- σ is conflictfree;
- $S_2 \cup \mathcal{A}_\sigma \models C \sqsubseteq D$ $(C \sqsupseteq D)$ for every assertion $C \sqsubseteq D \in \mathcal{A}$ $(C \sqsupseteq D \in \mathcal{A})$.

In Example 10, adding the assertion $\mathsf{HeartClinic} \equiv \mathsf{Heart}$ would only allow for the intuitive integration mapping from S_1 into S_2 (even together with the schemas defined in Example 5).

Of course, the whole point of our tool will be to prompt users with such assertions, rather than have users fish them out of thin air; but to the extent that such assertions become available serendipitously, they should be taken advantage off.

5 Limitations of Structural Knowledge

The number of possible integration mappings is closely related to the answers of the following questions: (i) How much structural knowledge is provided by a schema, or in other words, how much information lies merely in the intuition of the names? (ii) To what degree identifiers can be distinguished by their structure?

If there is only little structural information provided for identifiers by the schema, and thus, their is not much structural difference between the identifiers, then there are of course more conflictfree mappings than for schemas with rich structural information.

For example consider the following two schemas:

S_1 :

$$\begin{aligned} \mathsf{Patient} &\sqsubseteq \mathsf{Human} \\ \mathsf{Employee} &\sqsubseteq \mathsf{Human} \\ \mathsf{Patient} &\sqsubseteq \neg\mathsf{Employee} \end{aligned}$$

S_1' :

$$\begin{aligned} \mathsf{Patient} &\sqsubseteq \mathsf{Human} \\ \mathsf{Employee} &\sqsubseteq \mathsf{Human} \\ \mathsf{Patient} &\sqsubseteq \neg\mathsf{Employee} \\ \mathsf{Patient} &\sqsubseteq (\geq 1\,\mathsf{disease}) \end{aligned}$$

A total integration mapping from the set of identifiers of S_1 into itself can not only map Patient and Employee on themselves but it could also map Patient on Employee and Employee on Patient. There is no structural difference between these two concepts. On the other hand, in S_1' these concepts can structurally be distinguished. Therefore, only the first integration mapping, namely the identity mapping is valid for S_1'.

In this section, we will formalize some aspects of the limitations of the structural knowledge that is provided by a schema. For that purpose, we first formalize the notion of structurally indistinguishable identifiers and then generalize this notion to indistinguishable sets of identifiers.

5.1 Indistinguishable Identifiers

Intuitively, there is no structural difference specified above between Patient and Employee in S_1. In order to formalize this intuition, we need the following notation: For identifiers $i_1, \ldots, i_n, i_1', \ldots, i_n'$ and a schema S we define $S[i_1/i_1', \ldots, i_n/i_n']$ to be a schema that is obtained from S by simultaneously substituting i_j by i_j' for all $j = 1, \ldots, n$. Thus, for example, $S[A/B, B/A]$ means that A and B are switched in S.

Definition 12 Let S be a schema and let A, B be identifiers in S of the same type, i.e., both are concept names or both are roles. We call A and B *indistinguishable* in S ($A \approx_S B$ for short) if and only if $S \equiv S[A/B, B/A]$; otherwise A and B are called *distinguishable*.

According to this definition Parents and Employee are indistinguishable in S_1 but distinguishable in S_1'.

To state a simple characterization of indistinguishable identifiers we need the following notation: For identifiers $i_1, \ldots, i_n, i_1', \ldots, i_n' \in \Sigma$, where i_j is of the same type as i_j', and an interpretation $\mathcal{I}$ over the signature Σ we define $\mathcal{I}[i_1/i_1', \ldots, i_n/i_n']$ to be the interpretation that coincides with $\mathcal{I}$ on all identifiers distinct from $i_1, \ldots, i_n$ and that interprets $i_1, \ldots, i_n$ by ${i_1'}^I, \ldots, {i_n'}^I$, respectively.

Lemma 13 Let S be a schema and A, B be identifiers in S of the same type. Then, A and B are indistinguishable if and only if $\mathcal{I} \models S$ implies $\mathcal{I}[A/B, B/A] \models S$ for all interpretations $\mathcal{I}$.

PROOF. We first assume that A, B are indistinguishable. Now, let $\mathcal{I} \models S$. Since $S \equiv S[A/B, B/A]$ we know $\mathcal{I} \models S[A/B, B/A]$. But then $\mathcal{I}[A/B, B/A] \models S$.

For the if direction we assume that the right-hand side of the statement is true. Thus, $\mathcal{I} \models S$ implies $\mathcal{I}[A/B, B/A] \models S$. Therefore, $\mathcal{I} \models S[A/B, B/A]$. On the other hand, $\mathcal{I} \models S[A/B, B/A]$ implies $\mathcal{I}[A/B, B/A] \models S$. Then, by the assumption we know that $\mathcal{I} \models S$. This shows that $S \equiv S[A/B, B/A]$ which means $A \approx_S B$. ∎

As an easy consequence, we can show that $A \approx_S B$ is an equivalence relation.

Lemma 14 For a schema S the relation $\approx_S$ is an equivalence relation on the set of concept names/role names in S.

PROOF. Obviously, $\approx_S$ is reflexive and symmetric. It remains to be shown that $\approx_S$ is transitive. Let A, B, C be identifiers of S of the same type with $A \approx_S B$ and $B \approx_S C$. We need to show $A \approx_S C$. Let $\mathcal{I} \models S$. From $A \approx_S B$ and Lemma 13 we know that $\mathcal{I}[A/B, B/A] \models S$. Then, $B \approx_S C$ and Lemma 13 imply $(\mathcal{I}[A/B, B/A])[B/C, C/B] \models S$. Again, employing $A \approx_S B$ gives us $((\mathcal{I}[A/B, B/A])[B/C, C/B])[A/B, B/A] \models S$. Now, observe that $((\mathcal{I}[A/B, B/A])[B/C, C/B])[A/B, B/A] = \mathcal{I}[A/C, C/A]$. By Lemma 13 this shows that A and C are indistinguishable. ∎

In the example above, we have mapped Patient on Patient and Employee on Employee. As already mentioned, Patient $\approx_{S_1}$ Employee, i.e., Patient and Employee are mapped onto one equivalence class w.r.t. S_1. Furthermore, we have seen that one can switch the mapping of Patient and Employee, which results in mapping Patient on Employee and vice versa. In the following lemma we shall prove that switching equivalent identifiers in an integration mapping preserves conflictfreeness.

Lemma 15 Let S_1 and S_2 be two schemas with disjoint signatures. Furthermore, let A', B' be identifiers in S_1, A, B identifiers in S_2, and σ an integration mapping from the identifiers in S_1 into the identifiers in S_2 such that $\sigma(A') = A$, $\sigma(B') = B$ and $A \approx_{S_2} B$. Then, σ' obtained from σ by switching the image of A' and B', i.e., $\sigma'(A') = B$ and $\sigma'(B') = A$ is an integration mapping from S_1 into S_2.

PROOF. Let $\mathcal{I} \models S_1$. Since σ is conflictfree, we know that $\mathcal{I}$ can be extended to a model $\mathcal{I}'$ of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$. Since A, B do not occur in S_1 it follows $\mathcal{I}'[A/B, B/A] \models S_1$. Since by Lemma 6, σ is injective, it follows that in $\mathcal{A}_\sigma$ the concepts A and B only occur in the axioms $A' \equiv A$ and $B' \equiv B$. Thus, by definition of σ' we can conclude $\mathcal{I}'[A/B, B/A] \models \mathcal{A}_{\sigma'}$. Finally, $\mathcal{I}'[A/B, B/A] \models S_2$ since $A \approx_{S_2} B$. This shows that there is an extension $\mathcal{I}'[A/B, B/A]$ of $\mathcal{I}$ with $\mathcal{I}'[A/B, B/A] \models S_1 \cup S_2 \cup \mathcal{A}_{\sigma'}$.

Now, let $\mathcal{I} \models S_2$. Because $A \approx_{S_2} B$ we know $\mathcal{I}[A/B, B/A] \models S_2$. Furthermore, $\mathcal{I}[A/B, B/A]$ can be extend to a model $(\mathcal{I}[A/B, B/A])'$ of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$. As before, we obtain $(\mathcal{I}[A/B, B/A])'[A/B, B/A]) \models S_1 \cup S_2 \cup \mathcal{A}_{\sigma'}$. By construction, this model is an extension of $\mathcal{I}$. ∎

Closely related to the lemma stated above is the following fact.

Lemma 16 Let S_1 and S_2 be schemas with disjoint signatures Σ_1 and Σ_2, respectively. Let σ be an integration mapping Σ_1 into Σ_2. Furthermore, let $\sigma(A) = B$. Finally, let B' be an identifier in S_2 not contained in the image of σ and indistinguishable to B. Then the mapping σ' obtained from σ by mapping A on B' instead of B is an integration mapping.

PROOF. Let $\mathcal{I} \models S_1$. Then, there exists an extension $\mathcal{I}'$ of $\mathcal{I}$ model of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$. Since B and B' do not occur in S_1 it follows $\mathcal{I}'[B/B', B'/B] \models S_1$. By Lemma 13, $\mathcal{I}'[B/B', B'/B] \models S_2$. Moreover, by construction $\mathcal{A}_\sigma$ and $\mathcal{A}_{\sigma'}$ coincide except for $A \equiv B$ which in $\mathcal{A}_{\sigma'}$ is replaced by $A \equiv B'$. Since B, B' do not occur somewhere else in the assertions it follows $\mathcal{I}'[B/B', B'/B] \models \mathcal{A}_{\sigma'}$. Thus, $\mathcal{I}'[B/B', B'/B]$, which is an extension of $\mathcal{I}$, is a model of $S_1 \cup S_2 \cup \mathcal{A}_{\sigma'}$.

Now, let $\mathcal{I} \models S_2$. As $B \approx_{S_2} B'$, we know $\mathcal{I}[B/B', B'/B] \models S_2$. Furthermore, there exists an extension $(\mathcal{I}[B/B', B'/B])'$ of $\mathcal{I}[B/B', B'/B]$ which is a model of $S_1 \cup S_2 \cup \mathcal{A}_\sigma$. As above, we get $(\mathcal{I}[B/B', B'/B])'[B/B', B'/B] \models S_1 \cup S_2 \cup \mathcal{A}_{\sigma'}$. By construction, that model is an extension of $\mathcal{I}$. ∎

On the one hand, the two preceeding lemmas show that because of indistinguishable identifiers one can derive additional integration mappings from given ones. On the other hand, the lemmas also provide us with a way of representing sets of integration mappings from Σ_1 into Σ_2 in a more compact way. Instead of Σ_2 one can map into the set of equivalence classes $\Sigma_2/_{\approx_{S_2}}$. Then, if σ is given as a mapping into these classes one can derive other integration mappings as follows: For a class $[A] \in \Sigma_2/_{\approx_{S_2}}$ let $E \subseteq \Sigma_1$ be the set of all identifiers that are mapped onto $[A]$. New integration mappings can be computed by simply defining an injective mapping from E into $[A]$. This can be done for every equivalence class of Σ_2. By putting these injective mappings together one obtains new integration mappings.

It cannot structurally be determined which one of those derived mappings is the right one, i.e., the mapping that coincides with the intuition behind the identifiers. One has to use other informations, like semantic word distance of the names of the identifiers or input from the user, to decide which element in an equivalence class fits best.

It should be mentioned that for integration mappings σ, $A \approx_{S_1} B$ does not imply $\sigma(A) \approx_{S_2} \sigma(B)$. Example: Let S_1 consist of the axioms $A \sqsubseteq C$, $B \sqsubseteq C$ and S_2 of $A' \sqsubseteq C'$, $B' \sqsubseteq C'$, $D' \sqsubseteq A'$. Then, σ mapping A on A', B on B' and C on C' is an integration mapping. However, although $A \approx_{S_1} B$, A' and B' are not indistinguishable.

For that reason, mappings from equivalence classes of Σ_1 into equivalence classes of Σ_2 are not well-defined since the images of the elements of one equivalence class of Σ_1 are not necessarily equivalent w.r.t. S_2.

5.2 Indistinguishable sets of identifiers

In the previous section, we only related single identifiers to each other. Here we want to generalize these relationships in order to relate sets of identifiers that structurally cannot be distinguished from other sets of identifiers.

Before formally defining the generalized relationship, we illustrate the new notion by means of a simple example: Let S be the schema consisting of the axioms:

$$\begin{aligned}
\mathsf{Patient} &\sqsubseteq \forall \mathsf{disease}.\mathsf{Disease} \\
\mathsf{Heart\text{-}Patient} &\sqsubseteq \mathsf{Patient} \\
\mathsf{Employee} &\sqsubseteq \forall \mathsf{address}.\mathsf{Address} \\
\mathsf{Boss} &\sqsubseteq \mathsf{Employee} \\
\mathsf{Patient} &\sqsubseteq \neg \mathsf{Employee}
\end{aligned}$$

Obviously, switching the identifiers describing the patient and the ones for employees, i.e., switching Patient and Employee, disease and address, Disease and Address, Heart-Patient and Boss leads to an equivalent schema. We shall call these two sets indistinguishable.

For the formal definition we need some notation: Let S be a schema with signature Σ and let φ be a partial, injective mapping from Σ into Σ where the domain $\mathsf{domain}(\varphi)$ of φ and the image $\mathsf{image}(\varphi) := \varphi(\mathsf{domain}(\varphi))$ of φ are disjoint. Now, let $\{i_1, \ldots, i_n\}$ be an enumeration of the domain of φ. Then, we define $S[\varphi]$ to be the schema $S[i_1/\varphi(i_1), \varphi(i_1)/i_1, \ldots, i_n/\varphi(i_n), \varphi(i_n)/i_n]$. Thus, if the domain of φ is $\{A\}$ and $\varphi(A) = B$, then $S[A/B, B/A] = S[\varphi]$. In the same way, we generalize the definition of $\mathcal{I}[A/B, B/A]$ to $\mathcal{I}[\varphi]$ where $\mathcal{I}$ is an interpretation over Σ.

Definition 17 Let S be a schema with signature Σ. Furthermore, let φ be a partial, injective mapping from Σ into Σ such that the domain $\mathsf{domain}(\varphi)$ and the image $\mathsf{image}(\varphi)$ of φ are disjoint. Then, φ is called *indistinguishable* in S if and only if $S \equiv S[\varphi]$.

Note that as long as subsumption modulo a schema is decidable, it is decidable if φ is indistinguishable in S since in order to check $S \equiv S[\varphi]$ one only has to check whether every axiom in S is implied by $S[\varphi]$ and vice versa. Consequently, the set of all indistinguishable φ's can be computed.

Lemma 15 can be generalized as follows:

Lemma 18 Let S_1 and S_2 be two schemas with disjoint signatures Σ_1 and Σ_2, respectively, and σ be an integration mapping from Σ_1 into Σ_2. Furthermore, let φ be an indistinguishable mapping for S_2 such that there are two disjoint subsets M_1, M_2 of Σ_1 with $\sigma(M_1) = \mathsf{domain}(\varphi)$ and $\sigma(M_2) = \mathsf{image}(\varphi)$. Now, let σ' be a mapping obtained from σ with the following modification: for every $m \in M_1$ we define $\sigma'(m) := \varphi(\sigma(m))$ and for every $m \in M_2$ we specify $\sigma'(m) := \varphi^{-1}(\sigma(m))$. Then, σ' is an integration mapping from S_1 into S_2.

PROOF. Analogously to the proof of Lemma 15. ∎

The generalization of Lemma 16 can be stated as follows:

Lemma 19 Let S_1 and S_2 be two schemas with disjoint signatures Σ_1 and Σ_2, respectively, and σ be an integration mapping from Σ_1 into Σ_2. Furthermore, let φ be an indistinguishable mapping for S_2 such that $\mathsf{domain}(\varphi) \subseteq \mathsf{image}(\sigma)$ and $\mathsf{image}(\varphi) \cap \mathsf{image}(\sigma) = \emptyset$. Now, let σ' be a mapping that coincides with σ except that for every $A \in \Sigma_1$ with $\sigma(A) \in \mathsf{domain}(\varphi)$ we define $\sigma'(A) := \varphi(\sigma(A))$. Then, σ' is an integration mapping from S_1 into S_2.

PROOF. Analogously to the proof of Lemma 16. ∎

5.3 Indistinguishable Identifiers in a Real Ontology

In order to get a feel for the occurrence of indistinguishable identifiers, we have computed the equivalence classes of indistinguishable concept names and role names in the medical ontology GALEN. Lemma 15 and 16 show the impact of indistinguishable concepts on the number of integration mappings. Intuitively, for our structural integration approach we prefer ontologies with as few indistinguishable concepts as possible, in other words, with as much structure as possible.

The underlying description logic of the GALEN knowledge base is $\mathcal{FLE}$. The axioms are of the form $C \sqsubseteq D$ where both C and D are $\mathcal{FLE}$-concept descriptions. In addition, GALEN contains a role hierarchy, i.e., one allows for axioms of the form $r \sqsubseteq r'$ where r, r' are roles.

We have employed a correct but incomplete heuristic for computing the equivalence classes of indistinguishable identifiers. As a result, the actual equivalence classes might be bigger than the ones we computed. However, the heuristic, tailored to the GALEN ontology, seems to be a good approximation. Roughly speaking, our algorithm checks for two given identifiers whether switching these identifiers in the schema leads to a syntactically equal schema.

	number of identifiers in GALEN	number of identifiers in NTE	number of identifiers in the biggest class
concepts	2727	910	29
roles	413	116	9

Table 2: indistinguishable identifiers in GALEN

The results of our computation are depicted in Table 2 where NTE stands for "non-trivial equivalence classes" which refers to all those classes with more than one element.

On the one hand, the table shows that GALEN has a "rich" structure since two thirds of the concept names can be distinguished structurally from all other concepts in the ontology. On the other hand, indistinguishability plays a surprisingly large role in the ontology since one third of the concepts cannot be distinguished from all other concepts.

Among others, this means that the set of integration mappings from GALEN into GALEN does not contain only the identity mapping, which, in this case, is the intuitive mapping that one would expect to be computed; Lemma 15 and 16 show that one can derive other integration mappings from the identity mapping.

6 Algorithmic Issues

In this section, we examine the problem of computing integration mappings. First, we show that several versions of this problem are not "easy", and then specify a generic algorithm for computing integration mappings based on the notion of *unification* for description logics. Finally, we present some initial empirical results on the number of integration mappings in real ontologies.

6.1 Some NP-hardness Results

We show that deciding the existence of a total integration mapping is an NP-hard problem even for description logics, such as $\mathcal{FL}_0$, in which other reasoning (e.g., computing subsumption) is easy.

First we consider integration mappings that take identifiers to identifiers only, and show that this problem is related to 1-in-3-SAT, a known NP-hard problem [GJ79]. Let ϕ be a 1-in-3-SAT formula, i.e., a conjunction of n clauses c_i of the form $c_i = p \vee q \vee r$ where p, q, r are propositional variables. We define two $\mathcal{FL}_0$-schemas S_1 and S_2 over the signatures Σ_1 and Σ_2 as follows. For S_1 we introduce the concept names $C_1, \ldots, C_n$ as well as A_p for every propositional variable p. In addition, we need a role name s; S_2 is defined over the concept names D_i^1, D_i^2, D_i^3 for $i = 1, \ldots, n$ as well as G_p^0, G_p^1 for every propositional variable p. Again, we need one role t. The idea is that A_p is mapped on G_p^0 or G_p^1 corresponding to assigning p to false or true.

Now, for every clause $c_i = p \vee q \vee r$, S_1 contains the axiom

$$C_i \quad \sqsubseteq \quad \forall s^{3i}.A_p \sqcap \forall s^{3i}s.A_q \sqcap \forall s^{3i}ss.A_r$$

where s^m for some non-negative integer m is the word $s \cdots s$ of length m.

For every clause $c_i = p \vee q \vee r$, S_2 contains the axioms

$$\begin{aligned} D_i^1 &\sqsubseteq \forall t^{3i}.G_p^0 \sqcap \forall t^{3i}t.G_q^0 \sqcap \forall t^{3i}tt.G_r^1, \\ D_i^2 &\sqsubseteq \forall t^{3i}.G_p^0 \sqcap \forall t^{3i}t.G_q^1 \sqcap \forall t^{3i}tt.G_r^0, \\ D_i^3 &\sqsubseteq \forall t^{3i}.G_p^1 \sqcap \forall t^{3i}t.G_q^0 \sqcap \forall t^{3i}tt.G_r^0. \end{aligned}$$

Lemma 20 ϕ is satisfiable iff there exists a total integration mapping from Σ_1 into Σ_2.

PROOF. For the only-if direction of the lemma, let β be an assignment of the propositional variables satisfying ϕ. Now, we define a total mapping σ from Σ_1 to Σ_2 as follows: A_p is mapped on G_p^0 if $\beta(p) = \mathsf{false}$ and on G_p^1 otherwise; C_i is mapped on one of the concepts D_i^1,D_i^2, or D_i^3 corresponding to the mappings of the A_p's. Finally, s is mapped on t. It is easy to verify that σ is an integration mapping from S_1 into S_2.

Now, we prove the if direction. Since there are no axioms for the concept names A_p's, we know that $S_1 \models A_p \sqsubseteq E$ implies $E \equiv \top$. In S_2, this property is only satisfied by the concepts G_*^*. Thus, as an easy consequence of Lemma 7 the concepts A_p are mapped on some G_*^*. Then, because of the structure of role chains, it is not hard to see that C_i must be mapped on one of the concepts D_i^1, D_i^2, D_i^3. As a result, the A_p's are taken either to G_p^0 or to G_p^1 and for every $c_i = p \vee q \vee r$ only one of the concepts A_p, A_q, A_r is mapped on G_*^1, $* \in \{p, q, r\}$. Thus,

$$\beta(p) := \begin{cases} \mathsf{true} & \text{if} \quad \sigma(A_p) = G_p^1, \\ \mathsf{false} & \text{if} \quad \sigma(A_p) = G_p^0 \end{cases}$$

satisfies ϕ. ∎

As an immediate consequence of the lemma, we obtain

Proposition 21 Deciding the existence of total integration mappings from identifiers to identifiers for $\mathcal{FL}_0$-schemas is an NP-hard problem.

Note that in the proof we use only a constant number of role names — in fact, exactly one for each schema.

One can show NP-hardness even for integration mappings that can map identifiers onto complex expressions. However, the proof we present here requires an unbounded number of roles as well as predefined interschema assertions.

Again, the proof is by reduction to 1-in-3-SAT. For every clause $c_i = p \vee q \vee r$ the schema S_1 contains the axiom

$$A_i \quad \sqsubseteq \quad \forall R_p.\forall R_q.\forall R_r.B$$

and for every propositional variable p the axiom

$$A_p \quad \sqsubseteq \quad \forall R_p.B.$$

Furthermore, S_2 contains for every propositional variable p and clause $c_i = p \vee q \vee r$ the axioms

$$\begin{aligned} A_i' &\sqsubseteq \forall R_p^0.\forall R_q^0.\forall R_r^1.B \sqcap \forall R_p^0.\forall R_q^1.\forall R_r^0.B \sqcap \forall R_p^1.\forall R_q^0.\forall R_r^0.B \\ A_p' &\sqsubseteq \forall R_p^0.B' \sqcap \forall R_p^1.B'. \end{aligned}$$

Finally, we need the interschema assertions

$$\begin{aligned} A_p &\equiv A_p' \\ B &\equiv B' \\ A_i &\equiv A_i' \end{aligned}$$

for every propositional variable p and clause c_i. Thus, every integration mapping must map A_p on A_p', B on B' and A_i on A_i'. Because of the axioms to A_p and A_p', we know that every R_p is mapped

either on R_p^0 or on R_p^1, which corresponds to mapping p on false or true. Now, the axioms to A_i and A_i' ensure that the assignment of the propositional variables induced by an integration mapping satisfies ϕ. Analogously, every truth assignment for ϕ induces an integration mapping. As a result we obtain:

Proposition 22 Deciding the existence of total integration mappings w.r.t. interschema assertions for $\mathcal{FL}_0$-schemas is an NP-hard problem.

6.2 A Generic Algorithm for Computing Integration Mappings

In this section, we relate the problem of computing total integration mappings to that of finding solutions to certain "unification" problems involving descriptions.

Suppose we allow variable symbols to appear in descriptions, so they can be thought of as patterns that can be matched against other concepts (or even other patterns, resulting in a unification problem). Then, intuitively we can think of all the identifiers in S1 as variables, and the integration mapping should be one of the substitutions resulting from matching all the concepts in S1 against the concepts in S2.

6.2.1 Unification modulo schemas

The notion of matching and unification between concepts has been explored in [BM96, BKBM99, BN98].

Definition 23 Let N_C, N_R, N_C^X, and N_R^X be four disjoint sets, the set of concept names, the set of role names, the set of concept variables, and the set of role variables. The set of all $\mathcal{FL}_0$-concept patterns over the four sets is inductively defined as follows:

- Every concept name and concept variable is a concept pattern.
- The symbol $\top$ is a concept pattern.
- If C and D are concept patterns, then $C \sqcap D$ is a concept pattern.
- If C is a concept pattern and R is a role name or a role variable, then $\forall R.C$ is a concept pattern.

A *substitution* σ is a mapping from the set of concept and role variables into the set of $\mathcal{FL}_0$-concept *descriptions* and role chains, respectively. This mapping is extended to concept patterns in the obvious way, i.e.,

- $\sigma(A) := A$ for all $A \in N_C$,
- $\sigma(\top) := \top$,
- $\sigma(C \sqcap D) := \sigma(C) \sqcap \sigma(D)$, and
- $\sigma(\forall R.C) := \forall \sigma(R).\sigma(C)$ where $\sigma(R) := R$ in case $R \in N_R$.

Definition 24 Let S be a $\mathcal{FL}_0$-schema over N_C and N_R. Then a *unification problem modulo* S is of the form $S \gg \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ where C_i, D_i, $i = 1, \ldots, n$ are concept patterns over N_C, N_R, N_C^X, and N_R^X. A substitution σ is a *unifier* of this problem iff $S \models \sigma(C_i) \sqsubseteq \sigma(D_i)$ for all $i = 1, \ldots, n$. In this case, the unification problem is called *unifiable* or *solvable*. In the case when the D_i contain no variables, the unification is said to reduce to a *matching* problem.

For example, let A, B, C be concept names, R be a role name, X, Z be concept variables, and Y be a role variable. If $S := \{A \sqsubseteq \forall R.\forall R.B, B \sqsubseteq C\}$, then a possible unifier for the unification problem $S \gg \{X \sqsubseteq \forall Y.Z\}$ is $\sigma = \{X \mapsto A, Z \mapsto C, Y \mapsto RR\}$ since $S \models A \sqsubseteq \forall RR.C$.

6.2.2 The Generic Algorithm

In the sequel, let S_1 and S_2 be two $\mathcal{FL}_0$-schemas over disjoint signatures Σ_1 and Σ_2. As an easy consequence of Lemma 9, we obtain that every conflictfree mapping from S_1 into S_2 is a unifier of the following unification problem modulo S_2 over the set of concept and role names in Σ_2 and the concept and role variables in Σ_1, i.e., the concept and role names of S_1 are now considered to be variables:

$$S_2 \gg \{A \sqsubseteq C \mid A \sqsubseteq C \in S_1\} \cup \{A \equiv C \mid A \equiv C \in S_1\}.$$

Integration mappings w.r.t. interschema assertions $\mathcal{A}$ satisfy the above unification problem extended by:

$$\{C \sqsubseteq D \mid C \sqsubseteq D \in \mathcal{A}\} \cup \{C \sqsupseteq D \mid \ \in \mathcal{A}\}.$$

Note that C is a concept pattern and D is a concept description.

Therefore integration mappings can be obtained by, first, computing sets of unifiers of the above unification problem, and second, deciding conflictfreeness for these unifiers.

As an example for our generic algorithm, consider the schemas

S_1 : $\mathsf{HeartClinic} \sqsubseteq \forall \mathsf{patients\text{-}have}.\mathsf{HeartDisease}$
S_2 : $\mathsf{Heart} \sqsubseteq \forall \mathsf{patients}.\mathsf{Heart\text{-}Patients}$

The identifiers of S_1 are turned into variables and we obtain the unification problem:

$$S_2 \gg \{X_{\mathsf{HC}} \sqsubseteq \forall X_{\mathsf{ph}}.X_{\mathsf{HD}}\}$$

A possible unifier is $\sigma := \{X_{\mathsf{HC}} \mapsto \mathsf{Heart}, X_{\mathsf{ph}} \mapsto \mathsf{patients}, X_{\mathsf{HD}} \mapsto \mathsf{Heart\text{-}Patients}\}$ which also yields an integration mapping.

It should be noted that the unification problem introduced above can also be understood as a matching problem with side conditions modulo a schema. Side conditions are of the form $X \sqsubseteq C$ where X is a variable and C is a concept *pattern*. Such problems, although modulo empty schemas, have been introduced in [BKBM99] where side conditions have been considered in order to further restrict possible matchers. In our context, the interschema assertions $C \sqsubseteq D$ and $C \sqsupseteq D$ where C consists of variables only (namely, the identifiers in S_1) and D is a concept description over Σ_2 induce a matching problem since only the left hand-side contains variables. The side conditions (on variables A) are given by the axioms $A \sqsubseteq C$, $A \equiv C$ in S_1 where, again, C contains variables only.

So far, it is only known that deciding the solvability of unification problems in $\mathcal{FL}_0$ w.r.t. empty schemas is EXPTIME-complete [BN98]. Unification and matching problems modulo schemas have not been investigated yet.

Conflictfreeness is known to be undecidable in the context of the relation model [Con86] and for First Order Logic [EJ95]. It is an open problem whether (S_2-)conflictfreeness is undecidable for Description Logics. Thus, there are a lot of algorithmic problems to solve before the generic algorithm can be employed to compute integration mappings.

6.3 Integration Mappings of Real Ontologies — An Idealized Experiment

In this section, we investigate how much structure real ontologies provide to support the task of computing an intuitive integration mapping. For this purpose, we think of the following experiment: Given are two copies of the same schema where one schema has, say, English and the other schema has Chinese identifiers. Of course the "identity mapping", which maps the English identifiers to the corresponding Chinese ones, is the intuitive integration mapping. Now the question arises: How much do integration mappings resemble the identity mapping?

By Section 5, we know that even in this idealized situation we cannot expect to obtain exactly the identity mapping since one can derive new integration mappings due to indistinguishable identifiers.

However, in order to get some clue to our question for a real ontology, we have computed total integration mappings from GALEN into GALEN.

More precisely, we have computed a mapping φ, called *set-mapping* in the following, from the set N_C of concept names in GALEN to the power-set $2^{N_C/\approx}$ of equivalence classes in $N_C/\approx$ where $\approx$ is the equivalence relation defined by indistinguishable concept names (see Section 5). The mapping φ has the following property:

> If there is a total integration mapping from N_C into N_C that maps $A \in N_C$ onto $B \in N_C$, then the equivalence class $[B]_\approx$ of B is an element of $\varphi(A)$.

The reverse direction does not hold in general, i.e., if $[B]_\approx \in \varphi(A)$, then this does not mean that there exists a total integration mapping that maps A on B. Thus, $\varphi(A)$ is an upper bound for the concept names A can be mapped on. The reason is that the algorithm for computing φ uses conditions on total (conflictfree) integration mappings that are necessary for conflictfreeness but not sufficient.

The algorithm we propose, henceforth called *set-algorithm*, consists of three steps which gradually refine the set-mapping φ. In the sequel of this section, we will illustrate each step of the algorithm with respect to some given schema S with signature Σ and the set $N_C \subseteq \Sigma$ of concept names occurring in S where $\top$ is considered to be a concept name equivalent to the top-concept $\top$. We then present the results we have obtained for GALEN.

The first step of the algorithm uses necessary conditions that every total integration mapping must satisfy regarding the super-concept/sub-concept hierarchy. The concept hierarchy of S is a Hasse diagram for the subsumption relation $\sqsubseteq$, i.e., a (rooted) directed acyclic graph $G_S = \langle V, E, \top\rangle$ where N_C is the set of vertices, $E = \{(A, B) \in V \times V \mid S \models A \sqsupseteq B$ and there is no $C \in V$, $C \neq A$, $C \neq B$ such that $S \models A \sqsupseteq C$ and $S \models C \sqsupseteq B\}$ is the set of edges, and $\top$ the root. (We assume that V does not contain two equivalent elements. Otherwise G_S would not be acyclic. In case, there are two equivalent elements one can substitute one of the elements by the other one which would yield an equivalent schema.) Note that every node in G_S is reachable from the root $\top$.

The next lemma shows that every total integration mapping from S to S is an isomorphism from the hierarchy of S onto itself. For two rooted directed graphs $H_1 = \langle V_1, E_1, r_1\rangle$ and $H_2 = \langle V_2, E_2, r_2\rangle$ a mapping $\psi : V_1 \longmapsto V_2$ is an *isomorphism* from H_1 onto H_2 iff i) $\psi(r_1) = r_2$, ii) ψ is bijective, and iii) $(a, b) \in E_1$ iff $(\psi(a), \psi(b)) \in E_2$ for all $a, b \in V_1$.

Lemma 25 Every total integration mapping $\sigma : \Sigma \longrightarrow \Sigma$ from S into S is an automorphism on $G_S = \langle V, E\rangle$ when restricted to $N_C \subseteq \Sigma$.

PROOF. Since no two elements in V are equivalent, we can conclude by Lemma 6 that σ is an injective mapping from V into V, and thus, σ is bijective. Now, let $(A, B) \in E$. By Lemma 7,

we know $S \models \sigma(A) \sqsupseteq \sigma(B)$. Assume that there is a $C \in V$ different from A and B such that $S \models \sigma(A) \sqsupseteq C$ and $S \models C \sqsupseteq \sigma(B)$. Again, by Lemma 7, $S \models A \sqsupseteq \sigma^{-1}(C)$ and $S \models \sigma^{-1}(C) \sqsupseteq B$, which is a contradiction to $(A, B) \in E$. Thus, there is no such C. This means $(\sigma(A), \sigma(B)) \in E$. As σ is bijective, we can analogously conclude that $(\sigma(A), \sigma(B)) \in E$ implies $(A, B) \in E$. Since G_S is a rooted graph, we need to show $\sigma(\top) = \top$, which is an easy consequence of the fact that σ is conflictfree. ∎

The first step of our algorithm for computing φ is based on necessary conditions that are satisfied by graph isomorphisms. In order to specify these conditions, we need some more notation: For a rooted directed graph $H = \langle V, E, r\rangle$ and a node $v \in V$, $\mathsf{in\text{-}degree}(v) := |\{w \in V \mid (w, v) \in E\}|$ denotes the number of in-going edges of v. Analogously, $\mathsf{out\text{-}degree}(v) := |\{w \in V \mid (v, w) \in E\}|$ denotes the number of out-going edges of v. Furthermore, $\mathsf{levels}(v) := \{n \mid$ there exists a path from the root r to v of length $n\}$ is the set of levels of v.

Observation 26 For every isomorphism ψ from $H_1 = \langle V_1, E_1, r_1\rangle$ onto $H_2 = \langle V_2, E_2, r_2\rangle$ and $v \in V_1$ it is true that:

1. $\mathsf{out\text{-}degree}(v) = \mathsf{out\text{-}degree}(\psi(v))$,
2. $\mathsf{in\text{-}degree}(v) = \mathsf{in\text{-}degree}(\psi(v))$,
3. $\mathsf{levels}(v) = \mathsf{levels}(\psi(v))$, and
4. if $X := \{v_1, \ldots, v_l\}$ are the children of v and $Y := \{w_1, \ldots, w_l\}$ are the children of $\psi(v)$, then there exists a bijective mapping ψ' from X onto Y such that there exists an isomorphism from $H_{1|_{v_j}}$ onto $H_{2|_{\psi'(v_j)}}$ for all $j = 1, \ldots, l$ where $H_{1|_{v_j}}$ and $H_{2|_{\psi'(v_j)}}$ denote the subgraphs of H_1 and H_2 with roots v_j and $\psi'(v_j)$, respectively.

The algorithm depicted in Figure 1 computes for every $w \in V_2$ a set $\chi(w)$ such that if there exists an isomorphism ψ from H_1 onto H_2 with $\psi(v) = w$, then $v \in \chi(w)$. This property of the algorithm is an immediate consequence of the observations stated above.

Let $\chi(w) := \emptyset$ for every $w \in V_2$
For every $w \in V_2$ in post-order do
  Let $Y := \{w_1, \ldots, w_k\}$ be the set of children of w
  For every $v \in V_1$ do
    Let $X := \{v_1, \ldots, v_l\}$ be the set of children of v
    If
      $\mathsf{out\text{-}degree}(w) = \mathsf{out\text{-}degree}(v)$ $(\Rightarrow l = k)$ and
      $\mathsf{in\text{-}degree}(w) = \mathsf{in\text{-}degree}(v)$ and
      $\mathsf{levels}(w) = \mathsf{levels}(v)$ and
      there is a bijective mapping ψ' from X onto Y such that
        $v_i \in \chi(\psi'(v_i))$ for every $i = 1, \ldots, l$
    then $\chi(w) := \chi(w) \cup \{v\}$.

Figure 1: First step of the algorithm computing φ

By Lemma 25, the algorithm of Figure 1 also yields a set-mapping $\varphi := \chi^{-1}$ when applying it to $H_1 = H_2 = G_S$ where for $v \in V_1$, $\chi^{-1}(v) := \{w \mid v \in \chi(w)\}$.

However, we can easily get smaller sets $\chi(w)$, and thus, $\varphi(v)$. The sets $\chi(w)$ computed so far do not take the information of higher levels into account. More precisely, if w is a child of w' in H_2, then every element in $\chi(w)$ must be a child of a node $v \in \chi(w')$ in H_1. Therefore, the sets $\chi(w)$ computed by the algorithm in Figure 1 can be intersected with

$$\{v' \in V_1 \mid \text{ there exists a parent } w' \text{ of } w \text{ and } v \in \varphi(w') \text{ such that } v' \text{ is a child of } v.\}$$

One can easily implement an algorithm intersecting the sets $\chi(w)$ with these sets where the nodes of H_2 are traversed in pre-order to get as small sets $\chi(w)$ out of the intersection as possible.

The first two steps of our algorithm are illustrated in Figure 2. After traversing the hierarchy bottom-up the sets $\chi(w)$ for the leaves are $\{6, 7, 8\}$. This is because the structure of the higher levels has not been taken into account in the first step. By only looking at the leaves there is no structural difference between the three nodes 6,7, and 8. After passing down the informations of the higher levels in the second step of the algorithm, in our example we almost obtain the one-to-one mapping except for the nodes 6 and 7. In a schema containing only the super-concept/sub-concept information of the hierarchy, the nodes 6 and 7 are indisintuishable such that we cannot expect to obtain a one-to-one mapping merely based on that hierarchy.

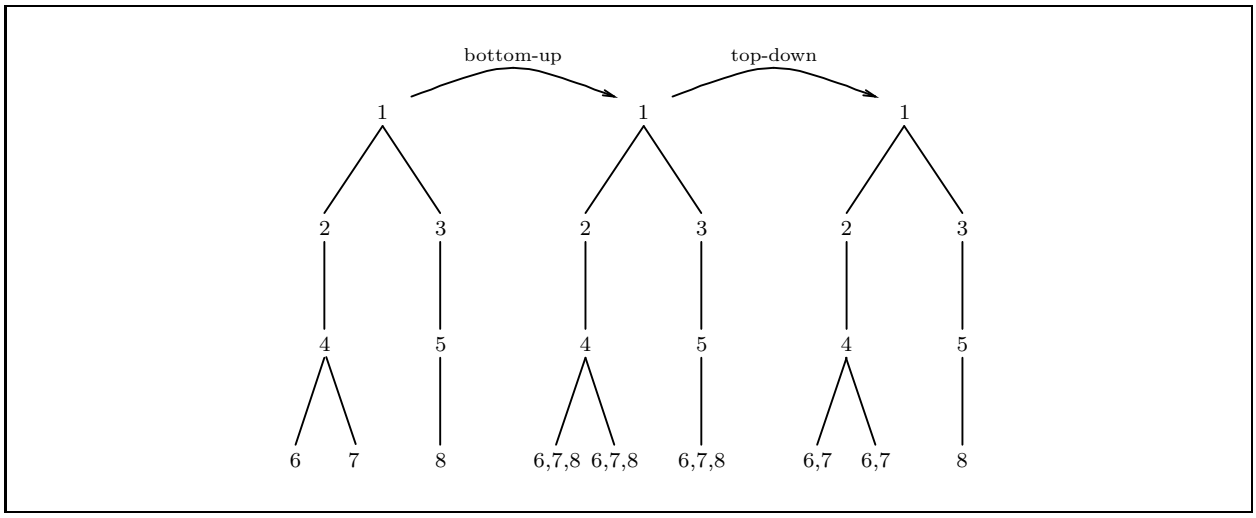

Figure 2: Illustrating the first two steps of the algorithm computing φ

Note that if the hierarchy is traversed top-down instead of bottom-up, one loses structural information from lower levels. Figure 3 shows that a second step passing information from the bottom two the top would yield better results, i.e., smaller sets $\chi(w)$.

Intuitively speaking, traversing the hierarchy bottom-up *and* top-down yields better results since one can use more information both from the bottom and from the top of the hierarchy.

In the third step of the refinement of $\varphi = \chi^{-1}$ we take the so-called local part-of hierarchy of concepts into account. Intuitively, this hierarchy represents the restrictions on roles and attributes of concepts.

Definition 27 The *local part-of hierarchy* of a concept name A in an $\mathcal{FLE}$-schema S consists of the set of in- and out-going edges of A.[1] A concept A has an *in-going edge* (B, A) iff there exists a role R such that

[1]We only consider $\mathcal{FLE}$-schemas here since GALEN is based on $\mathcal{FLE}$.

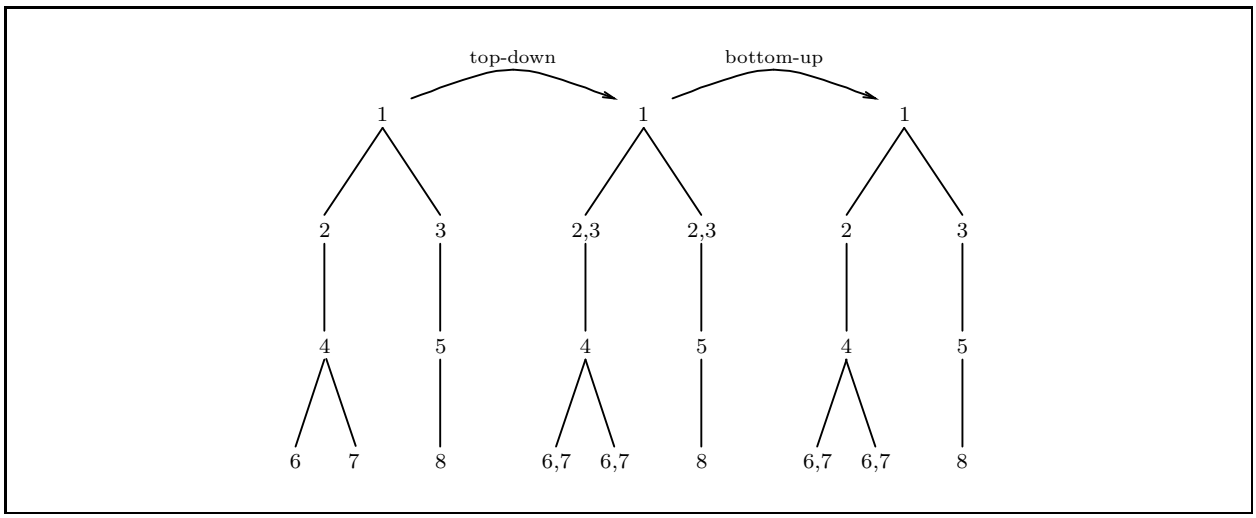

Figure 3: Traversing a hierarchy top-down

$$S \models B \sqsubseteq \exists R.A$$

where B is a concept name; A has an *out-going edge* $(A, \{B_1, \ldots, B_n\})$ iff there exists a role R such that

$$S \models A \sqsubseteq \exists R.(B_1 \sqcap \cdots \sqcap B_n)$$

where $\{B_1, \ldots, B_n\}$ is a maximal set (w.r.t. set inclusion) of concept names with that property.

So far, we know that a concept name A can only be mapped on the concepts in $\varphi(A)$. One can combine this information and the local part-of hierarchy to further cut down the set $\varphi(A)$.

The main observation for this step is the following: If A and D are concept names in S and A is in the first component of an in-going edge to D, then for every total integration mapping σ from the identifiers of S into itself, $\sigma(A)$ is a concept in an in-going edge to $\sigma(D)$. This is a consequence of the fact that $S \models A \sqsubseteq \exists R.D$ implies $S \models \sigma(A) \sqsubseteq \exists \sigma(R).\sigma(D)$ (cf. Lemma 7). Analogously, we can conclude that $\sigma(A)$ is a concept in some out-going edge of $\sigma(D)$ in case A is contained in an out-going edge of D.

Therefore, we know that A can only be mapped on the concepts in the following set $\pi(A)$ where $\mathsf{In}(D) := \{B \mid B \text{ is contained in some in-going edge of } D\}$ and $\mathsf{Out}(D) := \{B \mid B \text{ is contained in some out-going edge of } D\}$; note that $\bigcap_{\emptyset} \cdots := N_C$:

$$\bigcap_{A \in \mathsf{In}(D)} \bigcup_{B \in \varphi(D)} \{C \mid C \in \mathsf{In}(B)\} \cap \bigcap_{A \in \mathsf{Out}(D)} \bigcup_{B \in \varphi(D)} \{C \mid C \in \mathsf{Out}(B)\}.$$

Consequently, we know that a concept name A can only be mapped on $\varphi(A) := \varphi(A) \cap \pi(A)$ which is the output of the third and final step of our set-algorithm.

Applying this to the GALEN medical ontology with its 2727 concept names, we obtained the following results:

- 1727 concepts are mapped by φ on exactly one equivalence class.

- 2179 concepts are mapped by φ on at most two equivalence classes.

Since there are only 225 non-trivial equivalence classes, i.e., classes with more than one element, at least 1502 concepts can only be mapped on themselves and at least 1954 concepts can only be mapped on not more than two elements. This means that every total integration mapping from GALEN into GALEN maps more than one half of the concepts to the right element. Thus, using only the structural information of the concepts encoded in the GALEN ontology, one gets more than half of the the identity mapping which is the intuitive integration mapping in this case. One would expect the result to be even better if more of the structural information than the concept hierarchy and the local part-of hierarchy were used.

7 Summary and Future Work

Ontologies record the meaning of terms in some universe of discourse, by containing concept descriptions/definitions. We are interested in the problem of integrating several such ontologies that contain information about the same UofD. We are particularly concerned with ontologies described in descriptions logics, and containing large numbers of concepts.

After reviewing various approaches to information integration from the database literature, we have offered a formal framework for integration based on *inter-ontology assertions*, which relate concepts appearing in one ontology to (complex) descriptions expressible in the other ontology. A key property of such assertions is that they should not affect the meaning of the original ontologies (conflict-freeness [BC86]).

Our general goal is to explore the potential for generating automatically such IAs (as candidate integration assertions that could be proposed to users). Formally, this problem was characterized as one of finding "matches" between one ontology (treated as a pattern) and the other (treated as a target). Although the problem was proven to be intractable. We specify a relatively simple algorithm for finding IAs that are conflict free (and hence preserve the structural aspects of the terms being mapped to each other).

In empirical work, we considered the GALEN knowledge base, and tried to find automorphic integration mappings on it. Among others, we discovered that there are surprisingly many groups of "indistinguishable concepts" – ones whose structural properties are identical, and are therefore distinguished strictly by their identifiers. This is an interesting comment on the extent (actually lack of it) to which real ontologies provide "definitions" of their terms. To get around this problem, we group such indistinguishable concepts into equivalence classes, and only one member is used as a representative, in order to reduce the number of possible IAs that need to be considered by the user. (In a fully useful tool, a morphological and semantic analysis of the identifiers would also be used of course.) Furthermore, we found that over 50% of the (ideal) identity automorphism is recovered by our algorithm.

We are currently investigating additional heuristics for structure-based algorithms for finding conflict-free IAs. In the long term, we also want to integrate identifier-based heuristics. Of course, these heuristics need to be tested on real ontologies with large numbers of concepts, like GALEN and others.

Acknowledgements

This research was supported in part by the US National Science Foundation under grant IRI-9619979.

References

[ABM+89] A.Borgida, R. J. Brachman, D. L. McGuinness, , and L. A. Resnick. CLASSIC: a structural data model for objects. In *Proc. ACM SIGMOD'98*, pages 59–67, June 1989.

[BC86] Joachim Biskup and Bernhard Convent. A formal view integration method. In Carlo Zaniolo, editor, *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 28-30, 1986*, pages 398–407. ACM Press, 1986.

[BFT95] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: a preliminary report. In *1995 International Workshop on Description Logics (DL'95)*, Rome, Italy, June 1995. Also in the Proceedings of the International KRUSE Symposium, Santa Cruz, August 1995.

[BKBM99] F. Baader, R. Küsters, A. Borgida, and D. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.

[BLN86] C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database schema integration. *Computing Surveys*, 18(4):323–364, 1986.

[BM96] A. Borgida and D. L. McGuinness. Asking queries about frames. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, San Francisco, Calif., 1996. Morgan Kaufmann.

[BN98] F. Baader and P. Narendran. Unification of concept terms in description logics. In *Proceedings of the 13th biennial European Conference on Artificial Intelligence (ECAI-98)*. Brighton, UK, 1998.

[Bor95] Alex Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, October 1995.

[CCHJ94] J.J. Cimino, P.D. Clayton, G. Hripcsak, and S.B. Johnson. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *JAMIA*, 1(1):35–50, 1994.

[CDGL+98] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13, 1998.

[CGL+98] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[CGL99] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *IJCAI'99*, 1999.

[CL93] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Int. J. of Intelligent and Coorperative Information Systems*, pages 375–398, 1993.

[Con86] B. Convent. Unsolvable problems related to the view integration approach. In G. Ausiello and P. Atzeni, editors, *Lecture Notes in Computer Science*, volume 243, pages 141–156, Rome, Italy, September 1986. Springer.

[DH84] U. Dayal and H.-Y. Hwang. View definition and generalization for database integration in a multidatabase system. *IEEE TSE*, 10(6):628–645, 1984.

[EJ95] L. Ekenberg and P. Johannesson. Conflictfreeness as a basis for schema integration. In S. Bhalla, editor, *6th International Conference on Information Systems and Data Management*, pages 1–13, Bombay, 1995. Springer.

[FP93] C. Francalanci and B. Pernici. View integration: a survey of current developments. Internal Report 93-053, Politecnico di Milano, 1993.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[GLN92] W. Gotthard, P.C. Lockemann, and A. Neufeld. System-guided view integration for object-oriented databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(1):1–22, February 1992.

[Goh96] C. Goh. *Representing and Reasonning about Semantic Conflicts in Heterogeneous Information Systems*. PhD thesis, MIT Sloan School of Management, 1996. http://context.mit.edu/ coin/publications/.

[HPSar] I. Horrocks and P.F. Patel-Schneider. Optimizing description logic subsumption. *J. Logic and Computation*, (to appear).

[Hul86] R. Hull. Relative information capacity of simple relational database schemata. *SIAM J. Comput.*, 15(3):856–886, 1986.

[Kas97] Vipul Kashyap. *Information brokering over heterogeneous digita data: a medata-based approach*. PhD thesis, Rutgers University, Dept. of Computer Science, New Brunswick, 1997.

[LNE89] J.A. Larson, S.B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, April 1989.

[Mac87] R.M. MacGregor. A deductive pattern matcher. In *Proc. AAAI'87*, pages 403–408, 1987.

[MIR93] R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *VLDB'93*, pages 120–133, Dublin, Ireland, August 1993.

[MKSI96] E. Mena, V. Kashyap, A.P. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *CoopIS*, pages 14–25, 1996.

[RZStGC] A.L. Rector, P. Zanstra, W.D. Solomon, and the GALEN Consortium. Galen: Terminology services for clinical information systems. In *Health in the New Communications Age: Health care telematics for the 21st century*, pages 90–100, Amsterdam, ? IOS Press.

[SCC97] K.A. Spackman, K.E. Campbell, and R.A. Cote. Snomed rt: A reference terminology for health care. In *AMIA Annual Fall Symposium*, 1997.