ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

A. Formica

SIMILARITY OF XML-SCHEMA ELEMENTS SUPPORTED BY DOMAIN ONTOLOGIES

R. 631 Marzo 2005

Anna Formica – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, Viale Manzoni 30 - 00185 Roma, Italy. Email : anna.formica@iasi.cnr.it

This paper appears in Int. Journal of Software Engineering and Knowledge Engineering (IJSEKE) 15(1), pp.117-130, 2005.

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica, CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

XML-Schemas and domain ontologies are two key features of the Semantic Web. The possibility of assessing similarity between concepts is becoming a fundamental support in e-commerce and business transactions. In this paper a method for evaluating similarity of XML-Schema elements is proposed, by making use of domain ontologies. The method is inspired by a previous proposal concerning concept similarity within an enterprise ontology management system developed at IASI.

Keywords: *XML-Schemas, domain ontologies, similarity reasoning.*

1. Introduction

XML(eXtensible Markup Language)-Schemas and domain ontologies are two key features of the Semantic Web [6]. In particular, XML-Schemas [26] are acquiring a fundamental role since they allow data to be described across the internet. An XML-Schema provides a rich set of modeling constructors, types, and constraints, and is therefore expected to become the most common method for describing and validating highly structured XML documents [24].

A domain ontology is a "shared understanding of the domain of interest" [25], or a "formal, explicit specification of a shared conceptualisation" [15], where a conceptualisation is an abstract model of some phenomenon of the world which identifies the relevant concepts and relationships among concepts of that phenomenon. It is important to note that in this definition "shared" means that an ontology captures consensual knowledge, i.e., accepted by a panel of experts in the given domain, and "formal" refers to the fact that an ontology should be machine-understandable. Summarizing, a domain ontology contains a set of interrelated concepts (e.g., entities, attributes, etc.), each associated with a formal definition providing an unambiguous meaning of the concept in the given domain.

In the perspective of developing the Semantic Web, the existence of domain ontologies becomes fundamental. In fact, they allow XML tags to be associated with an unambiguous meaning in the given domain.

On the other hand, the possibility of assessing similarity between concepts, also referred to as *similarity reasoning* [18], is becoming a fundamental support in e-commerce and business transactions. In fact, when a certain good is not available with the desired characteristics, as for instance the price or the brand, a "similar" good can be acquired, although it is not exactly the one originally planned. Note that similarity reasoning is growing in importance in different areas, such as component-based information development, integration of multiple heterogeneous information sources for mediation and data warehousing etc..

In this paper a method for evaluating similarity of XML-Schema elements is proposed, by making use of domain ontologies. The method is inspired by a previous proposal concerning concept similarity within the enterprise ontology management system *SymOntos* developed at IASI [18]. The paper is organized as follows. In Section 2, the XML data model addressed in this work is presented via examples, and the proposed approach is informally introduced. Then, in Sections 3 and 4 the model and the method are formally presented, respectively. In particular, in Section 4, the notion of *element similarity* is formally introduced, by making use of the definition of *type similarity*. Then, the Related Work and Conclusion Sections follow.

2. The XML Data Model

In this section the XML data model addressed in this paper is presented, that focuses on a subset of the key modeling notions defined by *W3C* [26].

An XML-Schema is defined by a set of *element* and *type* declarations. An element is declared by a *name* and a *type*. Types, which can be named or unnamed, are simple (*simpleTypes*) or complex (*complexTypes*). SimpleTypes are atomic types as *string*, *decimal*, *integer*, *boolean*, *date* and *time*. ComplexTypes are declared by: (i) a sequence of elements, and/or (ii) a set of *attributes*, where an attribute is defined by a *name* and a simpleType.

Example 2.1. Consider the following declarations:

<xsd:element *name*="**accommodation**" *type*="**accommodationType**"/>

<xsd:complexType *name*="**accommodationType**">
  <xsd:sequence>
    <xsd:element *name*="**name**" *type*="**xsd:string**"/>
    <xsd:element *name*="**owner**" *type*="**personType**"/>
  </xsd:sequence>
  <xsd:attribute *name*="**country**" *type*="**xsd:string**"/>
</xsd:complexType>

In this example we have an element of name *accommodation*, whose type is *accommodationType*. It is a complexType defined by a sequence of two elements, the first is *name*, typed with the simpleType *string*, and the second is *owner*, typed with the complexType *personType* (which has to be defined), and one attribute of name *country*, typed with *string*. □

In this paper, for the sake of simplicity, a more compact syntax will be used, which will be formally defined in the next section. According to it, the element *accommodation* and the type *accommodationType* of the previous example become:

```
accommodation:accommodationType
accommodationType = <name:string,owner:personType>
    {country:string}
```

It is important to recall that in XML-Schemas recursive complexTypes (i.e., defined by types that, directly or indirectly, refer to the type being defined) are not allowed. In this paper both elements and attributes are assumed to be single-valued ($minOccurs = maxOccurs = 1$). Finally, in line with [13], in a complexType attributes with the same names are not allowed, whereas elements with the same names are allowed, subject to the restriction they have the same types.

ComplexTypes can also be defined in terms of other types, by organizing them according to a *type hierarchy*, by means of the *extension* and *restriction base* constructors. The types defined by using such constructors are referred to as *derived* types, whereas the types which are used to define the derived types are referred to as *base* types. Note that, according to [26], a complexType can be defined in terms of one base type at most, that is, by using the Object-Oriented terminology [2], in an XML-Schema multiple inheritance is not allowed. Notationally, the *complexContent* constructor is used, as shown in the example below.

Example 2.2. Consider the complexType *accommodationType* defined above. It is possible to define, for instance, the element *guestHouse* whose type is a derived complexType of name, for instance, *guestHouseType*, by adding elements and attributes to *accommodationType*, as follows:

<xsd:element *name*="**guestHouse**" *type*="**guestHouseType**"/>

<xsd:complexType *name*="**guestHouseType**">
  <xsd:complexContent>
    <xsd:extension base="**accommodationType**">
      <xsd:sequence>

```
            <xsd:element name="customer" type="studentType"/>
            <xsd:element name="dependent " type="personType"/>
          </xsd:sequence>
          <xsd:attribute name="price" type="xsd:integer"/>
          <xsd:attribute name="diningRoom" type="xsd:boolean"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

According to the compact syntax used in this paper, the above type becomes:

```
guestHouseType = ext(accommodationType) <customer:studentType,
    dependent:personType> {price:integer,diningRoom:boolean}
```

where **ext** stands for the *extension base* constructor.

Note that the complexType *guestHouseType* defined above is equivalent to its *type expansion*, i.e., the following complexType:

```
guestHouseType = <name:string,owner:personType,
       customer:studentType,dependent:personType>
       {country:string,price:integer,dinindRoom:boolean})
```

□

As an alternative to the extension base constructor, types can be organized according to a hierarchy by using the *restriction base* constructor. This constructor allows types to be derived by restricting the domains of already defined types. But, as opposed to the extension base constructor, in this case the derived type has not additional elements and/or attributes with respect to the base type [13]. For this reason, in this paper the restriction base constructor will not be addressed since, according to the similarity reasoning method proposed in the next sections, complexTypes having the same element and attribute names, typed with the same types or restrictions of them, behave "almost like" synonyms. For the same reason simpleTypes, which in XML-Schemas can also be organized according to a type hierarchy, will not be addressed.

2.1. The Method

The proposed method can be divided according to the following three phases.

Phase 1 - Acquisition of a domain ontology: the similarity graph
The first phase of the method is devoted to the acquisition of a domain ontology [15]. This step is necessary in order to assign an unambiguous meaning to the element names of the XML-Schema. According to [18], an ontology is a set of interrelated concepts, whose relations are *generalization* (*ISA*), *partOf*, *relatedness*, *similarity*, etc. In this paper, we focus on the similarity relation, which is a set of triples, each defined by two concepts (which are "similar") and a decimal number standing for their *axiomatic similarity degree*. The similarity relation can also be represented according to a *similarity graph*, i.e., a weighted undirected graph whose nodes are labeled with the concepts of the ontology and weights are the axiomatic similarity degrees.

Phase 2 - Type expansion
In order to evaluate type similarity, a necessary step is required which is referred to as *type expansion*. This step concerns the well-known problem of type *inheritance* related to the *generalization/specialization* relation [7, 17]. In our XML data model, it consists in the replacement of the extension base constructor of derived types with the elements and attributes of their base types, along the type hierarchy. An example of type expansion has been informally shown in the case of *guestHouseType* in Example 2.2. Once the expansion process has been performed, complexTypes are completely identified by the sequence of their elements and the set of their attributes and, therefore, they are ready to be compared.

Phase 3 - Evaluating element similarity
The axiomatic similarity degrees and the expanded complexTypes allow the computation of: (i) *type similarity* (ts) by analyzing the sequence of elements and the set of attributes of the expanded complexTypes (note that, with respect to [18] where only concept names are considered, here similarity is evaluated by taking into account not only the axiomatic similarity degrees of element and attribute names, but also the similarity of their related types); (ii) *element similarity* (es) which is given by the weighted average between the axiomatic similarity degree of the element names, as defined in the similarity graph, and the type similarity of the associated types.

3. Formal Basis

In the following $\mathcal{T}$, $\mathcal{E}$, and $\mathcal{A}$ are disjoint and countable sets of type names, element names and attribute names, respectively. Below the compact syntax adopted in this paper is formally defined.

A *type* of name τ is a declaration defined as follows:

$\tau = \prec tp \succ \mid \mathbf{ext}(\sigma) \prec tp \succ$
$\prec tp \succ ::= < e :\prec \gamma \succ [, e :\prec \gamma \succ] >$
$\mid \{ a :\prec \beta \succ [, a :\prec \beta \succ] \}$
$\mid < e :\prec \gamma \succ [, e :\prec \gamma \succ] > \{ a :\prec \beta \succ [, a :\prec \beta \succ] \}$
$\prec \gamma \succ ::= \lambda \mid \prec \beta \succ$
$\prec \beta \succ ::= \mathbf{string} \mid \mathbf{integer} \mid \mathbf{decimal} \mid \mathbf{boolean} \mid \mathbf{date} \mid \mathbf{time}$

where square brackets stand for iteration (zero, one or more), τ, σ, $\lambda \in \mathcal{T}$, $e \in \mathcal{E}$, and $a \in \mathcal{A}$. Furthermore, τ does not contain *name conflicts*, i.e.:

$e_h = e_k \Rightarrow \gamma_h = \gamma_k$, for all h,k, and $a_i \neq a_j$ for all i,j.

Atomic types, e.g. *string* or *integer*, are also referred to as *simpleTypes*, whereas type declarations, e.g. that of name τ, are referred to as *complexTypes*. In the case of the **ext** construct, the type σ is referred to as the *base type* of τ, and τ is a *derived type*. Recursive types are not allowed.

Let $\mathcal{B}$ be the set of atomic types. An *element* (of name) $e \in \mathcal{E}$ is a declaration of the form "e:τ", where $\tau \in \mathcal{T} \cup \mathcal{B}$.

Therefore, an XML-*Schema* Σ is a 5-uple:

$\Sigma = (T, E, A, T_{dec}, E_{dec})$

where $T \subset \mathcal{T}$, $E \subset \mathcal{E}$, $A \subset \mathcal{A}$, and T_{dec}, E_{dec} are sets of (declarations of) types and elements, respectively, with names in T, E, A, such that: (i) every type name in T has exactly one type declaration in T_{dec} and (ii) every type name in T_{dec} and E_{dec} is in T.

The *type hierarchy* of Σ is defined in line with the notion of a *signature for inheritance* [1, 3],

and it is a directed tree of root *anyType* (the most general type), indicated as $\mathcal{H}_\Sigma$, which relates derived types and base types.

Example 3.1. Consider the following sets of element and type declarations, which include the elements and types of Examples 2.1, 2.2. The type hierarchy of the XML-Schema defined starting from these sets of declarations is represented in Figure 1 (this will be the running example of this paper).

E_{dec} = {
accommodation:accommodationType
person:personType
guestHouse:guestHouseType
student:studentType
hotel:hotelType
grandHotel:grandHotelType
ruralHouse:ruralHouseType
farmHouse:farmHouseType
}

T_{dec} = {
accommodationType = <name:string,owner:personType>
{country:string}
personType = <firstName:string,lastName:string>
{address:string}
guestHouseType = **ext**(accommodationType) <customer:studentType,
dependent:personType> {price:integer,diningRoom:boolean}
studentType = **ext**(personType) <university:string>
hotelType = **ext**(accommodationType) <tourist:personType>
{swimmingPool:boolean,restaurant:boolean}
grandHotelType = **ext**(hotelType) {limoService:boolean}
ruralHouseType = **ext**(guestHouseType) {court:boolean}
farmHouseType = **ext**(guestHouseType) {dairy:boolean}
}
□

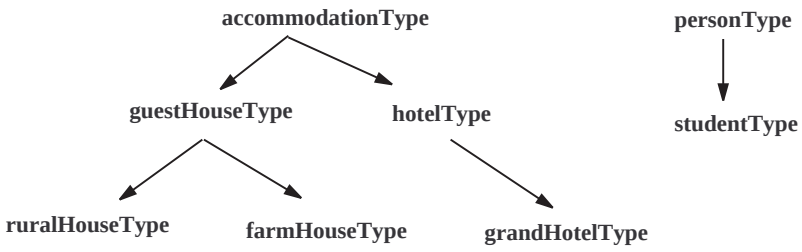

Figure 1: Type Hierarchy of Example 3.1

In the following, the *structural component* of a complexType τ:
$\tau = \mathbf{ext}(\sigma) < e_i : \gamma_i > \{a_j : \lambda_j\}$, $i = 0...n$, $j = 0...m$,
indicated as τ_c, is defined by the pair $\tau_c = (E(\tau), A(\tau))$ where:
$E(\tau) = < e_i : \gamma_i >$, $i = 0...n$
$A(\tau) = \{a_j : \lambda_j\}$, $j = 0...m$
and in the case $i = 0$ ($j = 0$), $E(\tau) = \emptyset$ ($A(\tau) = \emptyset$).

Analogously, the *structural component* of an XML-Schema Σ, indicated as Σ_c, is the schema Σ where all complexType declarations are replaced by their structural components.

3.1. Acquisition of a domain ontology: the similarity graph

As informally summarized in the previous section, the first phase of the method for evaluating element similarity consists in the acquisition of a *domain ontology*. According to [18], a *domain ontology* is specified by a set of concepts related by means of semantic relations, such us *generalization* (*ISA*), *partOf*, *relatedness*, *similarity* etc.. In particular, each pair of *similar* concepts (i.e., related by *similarity*) is associated with a decimal number in the interval [0.0 ... 1.0] standing for their *axiomatic similarity degree* (*axiomatic similarity* for short), which is established by means of a *Consensus System* [22] by a panel of experts in the given domain.

Given a domain ontology O, let Γ_O be the set of triples $\langle c_i, c_j, as(c_i, c_j)\rangle$ where c_i,c_j are similar concepts of the ontology, and $as(c_i, c_j)$ is their axiomatic similarity. Furthermore, we assume that for any concept c of the ontology $as(c, c) = 1$, and for any pair of concepts c_i, c_j of the ontology for which the axiomatic similarity is not defined $as(c_i, c_j) = 0$. Note that *similarity* is symmetric and for any pair of concepts c_i, c_j of the ontology $as(c_i, c_j) = as(c_j, c_i)$. Γ_O is referred to as the *similarity graph* of the ontology O.

Example 3.2. Consider our running example, and suppose to acquire an ontology O where all the element and attribute names of Example 3.1 are concepts defined in O, and whose similarity graph Γ_O contains the following triples:

$\{\ \langle hotel, accommodation, 0.8\rangle,$
$\langle hotel, grandHotel, 0.9\rangle,$
$\langle accommodation, grandHotel, 0.7\rangle,$
$\langle guestHouse, grandHotel, 0.6\rangle,$
$\langle hotel, guestHouse, 0.7\rangle,$
$\langle guestHouse, accommodation, 0.5\rangle,$
$\langle tourist, customer, 0.9\rangle,$
$\langle diningRoom, restaurant, 0.8\rangle,$
$\langle cost, price, 0.9\rangle\ \} \subset \Gamma_O$

Note that for the sake of simplicity all the triples with axiomatic similarity degrees equal to 0 and 1 have been omitted. This similarity (sub)graph is graphically shown in Figure 2.

□

An *enriched structural component* of an XML-Schema Σ is a triple $\mathcal{S} = (\Sigma_c, \mathcal{H}_\Sigma, \Gamma_O)$, where Σ_c is the structural component of Σ, $\mathcal{H}_\Sigma$ is the type hierarchy of Σ, and Γ_O is the similarity graph of an ontology O in the domain of the schema Σ.

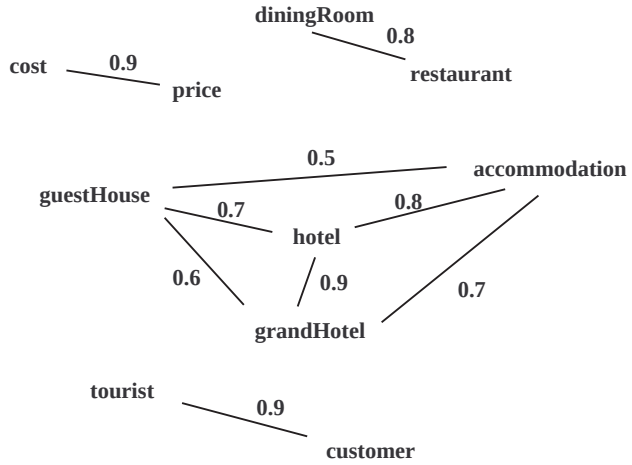

Figure 2: Similarity Graph

Example 3.3. Let Σ be the schema of our running Example. Then, an enriched structural component of Σ is the triple $\mathcal{S} = (\Sigma_c, \mathcal{H}_\Sigma, \Gamma_O)$, where $\mathcal{H}_\Sigma$ is the type hierarchy shown in Figure 1, Γ_O is the similarity graph represented in Figure 2, and Σ_c is the structural component of the schema where, for instance, in the case of *guestHouseType* we have:

$guestHouseType_c$ = (
$E(guestHouseType)$ = <`customer:studentType,dependent:personType`>,
$A(guestHouseType)$ = {`price:integer,diningRoom:boolean`})

□

3.2. Type Expansion

Type expansion concerns the well-known problem of *inheritance*, widely investigated in the literature, see for instance [9, 10]. This is a necessary step for computing element similarity, since we need to compare the structural components of their associated complexTypes, including all the elements and attributes of the ancestors of the types, up in the type hierarchy.

In the following, an expanded and enriched XML-Schema (EEXML-Schema) is an enriched structural component of an XML-Schema, indicated as $\mathcal{S}' = (\Sigma'_c, \mathcal{H}_\Sigma, \Gamma_O)$, where all derived complexTypes have been expanded (Σ'_c is the structural component of the schema where each complexType τ has been replaced with its expansion τ').

Example 3.4. For instance, the structural component of the expansion of the derived type *ruralHouseType* is the following:

$ruralHouseType'_c$ = (
$E(ruralHouseType')$ = <`name:string,owner:personType,`
`customer:studentType,dependent:personType`>
$A(ruralHouseType')$ = {`country:string,price:integer,`
`dinindRoom:boolean,court:boolean`})

□

4. Evaluating Element Similarity

In this section the following two notions of similarity will be formally introduced: (i) *type similarity*, which is recursively defined on types, on the basis of the axiomatic similarity of the element names and attribute names of type declarations; and (ii) *element similarity*, obtained by considering the axiomatic similarity of element names and the similarity of their related types.

4.1. Type Similarity

The notion of *type similarity* (ts) presented in this subsection is a revisitation of the *flat structural similarity* (fss) of concepts presented in [18], which has been modified to deal with XML-Schema complexTypes. It has been inspired by the *maximum weighted matching* problem in bipartite graphs, that can be solved in polynomial time [19].

In the following let $elemOf(\tau)$ and $attrOf(\tau)$ be the multiset and the set of element and attribute names of the the type τ, respectively (the symbol $*$ denotes a multiset [16]) - we recall that the multiset has been introduced since elements with the same names and the same types are allowed in XML-Schemas.

Consider two complexTypes of names τ_i,τ_j, and $attrOf$ (the following can be trivially extended to multisets in the case of $elemOf$). Within the cartesian product $attrOf(\tau_i) \times attrOf(\tau_j)$, consider all the sets of pairs such that there are no two pairs in the set sharing an element. Such sets will be referred to as *candidate sets of pairs*. For instance, assume that $attrOf(\tau_i)$ and $attrOf(\tau_j)$ represent a set of boys and a set of girls, respectively, a candidate set of pairs defines a possible set of marriages (when polygamy is not allowed) [19]. Successively, for each candidate set of pairs, consider the sum of the axiomatic similarity degrees (as) of the pairs. Then, the candidate set having the maximal among all the computed sums is chosen.

Once the candidate set and multiset of pairs with maximal sums have been chosen for both $attrOf$ and $elemOf$, respectively, ts is computed on the basis of the as of the selected pairs of element and attribute names, and the ts of their associated types.

Formally, consider two complexTypes τ_i,τ_j of an EEXML-Schema and let $\mathcal{R} \in \{elemOf, attrOf\}$. Let $n_\mathcal{R}$, $m_\mathcal{R}$ be the cardinalities of $\mathcal{R}(\tau_i)$, $\mathcal{R}(\tau_j)$, respectively, i.e., $n_\mathcal{R} = |\mathcal{R}(\tau_i)|$, $m_\mathcal{R} = |\mathcal{R}(\tau_j)|$, and suppose that $n_\mathcal{R} \leq m_\mathcal{R}$.

Then, the *set* $\mathcal{C}_\mathcal{R}(\tau_i, \tau_j)$ *of candidate sets/multisets of pairs* is defined by all possible sets/multisets of $n_\mathcal{R}$ pairs of names as follows [18]:

$$\mathcal{C}_\mathcal{R}(\tau_i, \tau_j) = \{\ \{\langle a_1, b_1\rangle \ldots \langle a_{n_\mathcal{R}}, b_{n_\mathcal{R}}\rangle\}^* \mid a_h \in \mathcal{R}(\tau_i),\ b_h \in \mathcal{R}(\tau_j),$$
$$\text{for all } h = 1 \ldots n_\mathcal{R}, \text{ and } a_h \not\equiv a_k,\ b_h \not\equiv b_l, \text{ for all } k,l \neq h\}.$$

Below the definition of ts follows. With respect to the notion of fss given in [18], where only the similarity of concept names has been considered, here the similarity of the types associated with the element and attribute names of the type declarations has also been addressed.

Definition 4.1. [Type similarity (ts)] Consider an EEXML-Schema $\mathcal{S}' = (\Sigma'_c, \mathcal{H}_\Sigma, \Gamma_O)$, and two types τ_i,τ_j of the schema. Let $\mathcal{O} = \{elemOf, attrOf\}$, $\mathcal{R} \in \mathcal{O}$, and $P_\mathcal{R} \in \mathcal{C}_\mathcal{R}(\tau_i, \tau_j)$ the candidate set/multiset of pairs having maximal sum, i.e. such that:

$$\sum_{\langle a,b\rangle \in P_\mathcal{R}} as(a,b) = \max_{P \in \mathcal{C}_\mathcal{R}(\tau_i,\tau_j)} \left(\sum_{\langle a,b\rangle \in P} as(a,b) \right)$$

Then, the *type similarity* of τ_i,τ_j, indicated as $ts(\tau_i, \tau_j)$, is defined as follows:

1. τ_i, τ_j are (names of) complexTypes. Then:

$$ts(\tau_i, \tau_j) = \sum_{\mathcal{R} \in \mathcal{O}} \left[\frac{w_{\mathcal{R}}}{m_{\mathcal{R}}} \sum_{\langle a,b \rangle \in P_{\mathcal{R}}} as(a,b) * ts(typeOf(\tau_i, a), typeOf(\tau_j, b)) \right]$$

2. τ_i or τ_j is a simpleType. Then:

$ts(\tau_i, \tau_j) = 1$ if $\tau_i = \tau_j$
$ts(\tau_i, \tau_j) = 0$ otherwise

where $typeOf(\tau_i, a)$, $typeOf(\tau_j, b)$ are the types associated with a,b, in the type declarations τ_i, τ_j, respectively, $m_{\mathcal{R}}$ is defined as above, $as(a, b)$ is the axiomatic similarity of a,b, according to the similarity graph Γ_O, and $w_{\mathcal{R}}$ are weights such that $\sum_{\mathcal{R} \in \mathcal{O}} w_{\mathcal{R}} \leq 1$. □

Note that $0 \leq ts(\tau_i, \tau_j) \leq 1$, and in the case of types with disjoint domains (e.g., *integer* and *boolean*, or *integer* and *personType*) similarity is equal to zero (whereas in the case of the same types similarity is equal to one). Finally, we want to remark that the weights $w_{\mathcal{R}}$ have been introduced to give more relevance to attributes rather than elements, or viceversa, to enrich the flexibility of the system.

Example 4.1. Consider our running example. Assume that, for both $elemOf$ and $attrOf$, $w_{\mathcal{R}} = \frac{1}{2}$ (i.e., $w_{elemOf} = w_{attrOf} = \frac{1}{2}$). Consider the complexTypes *ruralHouseType* and *grandHotelType*. The following holds:

$ts(ruralHouseType, grandHotelType) =$
$\frac{1}{2}\frac{1}{4}\ (1 + 1 + 0.9 * ts(studentType, personType)) +$
$\frac{1}{2}\frac{1}{4}\ (1 + 0 + 0.8 + 0)$

where:

$ts(studentType, personType) =$
$\frac{1}{2}\frac{1}{2}\ (1 + 1) + \frac{1}{2}\frac{1}{2}\ (1) = 0.75$

and, therefore:

$ts(ruralHouseType, grandHotelType) =$
$\frac{1}{2}\frac{1}{4}\ (1 + 1 + 0.9 * 0.75) + \frac{1}{2}\frac{1}{4}\ (1 + 0 + 0.8 + 0) = 0.55$

In this case the candidate sets of pairs with maximal sum are:

$\{\langle name, name\rangle, \langle owner, owner\rangle, \langle customer, tourist\rangle\}$
$\in \mathcal{C}_{elemOf}(ruralHouseType, grandHotelType)$
$\{\langle country, country\rangle, \langle price, swimmingPool\rangle, \langle diningRoom, restaurant\rangle,$
$\langle court, limoService\rangle\} \in \mathcal{C}_{attrOf}(ruralHouseType, grandHotelType)$.

Consider now the complexTypes *ruralHouseType* and *farmHouseType*. In this case:

$ts(ruralHouseType, farmHouseType) =$
$\frac{1}{2}\frac{1}{4}\ (1 + 1 + 1 + 1) + \frac{1}{2}\frac{1}{4}\ (1 + 1 + 1 + 0) = 0.87$

□

4.2. Element Similarity

The notion of *element similarity* (es) is given below. It is essentially given by the weighted average between the axiomatic similarity (as) of element names, as defined in the similarity graph, and the type similarity of their associated types.

Definition 4.2. [**Element similarity (es)**] Consider an EEXML-Schema $\mathcal{S}' = (\Sigma'_c, \mathcal{H}_\Sigma, \Gamma_O)$ and two element declarations of the schema, "$e_i : \tau_i$", and "$e_j : \tau_j$". Then, the *element similarity* of e_i,e_j, indicated as $es(e_i, e_j)$, is defined as follows:

$$es(e_i, e_j) = as(e_i, e_j) * p + ts(\tau_i, \tau_j) * (1 - p)$$

where $as(e_i, e_j)$ is the axiomatic similarity of e_i,e_j, defined according to Γ_O, $ts(\tau_i, \tau_j)$ is the type similarity of τ_i,τ_j, and p is a weight such that $0 \leq p \leq 1$.

□

Example 4.2. For instance, consider the *ruralHouse* and *grandHotel* elements of our running example and assume $p = \frac{1}{2}$. We have seen that:
$ts(ruralHouseType, grandHotelType) = 0.55$.
Since the axiomatic similarity for *ruralHouse* and *grandHotel* is null, their element similarity is defined as follows:
$es(ruralHouse, grandHotel) = \frac{0+0.55}{2} = 0.27$.
Whereas, with the same assumptions of the previous example, in the case of *accommodation* and *guestHouse*, we have:
$ts(accommodationType, guestHouseType) =$
$\frac{1}{2}\frac{1}{4}\ (1 + 1) + \frac{1}{2}\frac{1}{3}\ (1) = 0.41$
$as(accommodation, guestHouse) = 0.5$
therefore:
$es(accommodation, guestHouse) = \frac{0.5+0.41}{2} = 0.45$

□

5. Related Work

In the literature, a lot of work has been developed regarding the modeling capabilities and the expressive power of XML-Schemas (see, for instance, [21, 27, 24]). On the other hand, similarity reasoning has been tackled in different fields of *Computer Science*, although the large majority of results have not been conceived in e-commerce and business-to-business interoperability contexts, but rather in data integration for distributed query processing and/or data warehousing [11, 20]. Below the main differences among the approach presented in this paper and the majority of existing proposals, including [18] which has inspired this work, are mentioned. However, note that, to our knowledge, there are no proposals in the literature concerning similarity reasoning methods for XML-Schemas, which is the focus of this paper.

First of all, it is important to note that in [18] similarity has been evaluated according to the partitioning of the structural definition of a concept into different slots, which are *attributes* (Pr), *parts* (Pa), and *related* (R) concepts, comparing therefore only concept definition components which belong to the same partitions. In this paper, this methodology has been adopted and similarity of complexTypes has been addressed by partitioning their structural components into two slots, one related to elements and the other one related to attributes. Conversely, the majority of similarity reasoning methods found in the literature consider one kind of slot only, and in particular attributes.

An aspect that it is worth mentioning in comparing the different metrics proposed in the literature concerns hierarchically related concepts (types). Within *Semantic nets* and logic-based *Knowledge Representation*, we recall [23], where a metric on the power set of nodes in a semantic

net has been proposed. In particular, the conceptual distance of concepts that are hierarchically related has been defined by considering the length of the shortest path connecting them. Furthermore, in [8] the *Semantic-Distance Metric* (SDM) has been defined, which is based on weighted paths. In particular, in that paper concepts are connected by hyperonym/hyponym and synonym links, but concepts with similarity degrees strictly lesser than one are not addressed. In [11, 14], a constant value (specifically 0.5) is associated with *any* pair of hierarchically related concepts, independently of the level of refinement of the concepts - which is essentially related to the number of their common *properties* (elements and attributes, in the case of XML complexTypes). In [18], the notion of *hierarchical structural similarity* has been defined. Such a notion is based on the *extensional* aspect of inheritance, i.e., on the distribution of concept instances along the hierarchy. In particular, it is related to the degree of refinement between concepts: the greater is the refinement, the higher is the distance between the concepts. In this paper, since XML-Schema complexTypes cannot be directly instantiated, similarity of hierarchically related types has been evaluated on the basis of the *intensional* notion of inheritance, i.e., their expanded structural components, solution which, at the same time, reflects the level of refinement of types.

Concerning the general notion of concept similarity, we did not adopt the popular *Dice*'s function, as for instance in [4, 11]. In fact, with respect to our approach, such a function introduces a simplification since concept similarity is evaluated on the basis of the number of similar concept components divided by the total number of concept components of the two concepts, without explicitly considering in the computation their similarity degree. Analogously, in [12] semantic relatedness (similarity) evaluation is based on the aggregation of the interconnections between concepts, that is, the more properties two concepts have in common, the more closely related they are. Finally, in [5], general forms of distance metrics for the computation of similarity measures have been defined, although with more emphasis on the evaluation of the similarity between instances, rather than concepts.

6. Conclusion

The coexistence of domain ontologies and XML is fundamental in the perspective of the development of the Semantic Web [6]. In this paper an existing method for similarity reasoning in the context of domain ontologies has been revisited and extended to deal with XML-Schema elements.

As a future work, we plan to extend the proposed method to a richer XML data model, including for instance the *choice* and *union* constructors.

References

[1] H.Ait-Kaci, R.Nasr; *LOGIN: A Logic Programming Language with Built-In Inheritance*; Journal of Logic Programming (JLP), 3(3), 185-215, 1986.

[2] C.Beeri; *A formal approach to object-oriented databases*; Data & Knowledge Engineering 5; 353-382; North-Holland, 1990.

[3] C.Beeri, A.Formica, M.Missikoff; *Inheritance Hierarchy Design in Object-Oriented Databases*; Data & Knowledge Engineering, 30(3), 191-216, 1999.

[4] Bergamaschi, S. and Castano, S. and De Capitani di Vimercati, S. and Montanari, S. and Vicini, M. (1998) An Intelligent Approach to Information Integration. In Guarino, N. (ed), Formal Ontology in Information Systems. IOS Press, Amsterdam.

[5] Bisson, G. (1992) Learning in FOL with a similarity measure. Proceedings of 10th National Conference on Artificial Intelligence, San Jose, CA, July 12-16, 82-87, The AAAI Press/The MIT Press, CA.

[6] T. Berners-Lee et al; *The Semantic Web*; Scientific American, May 2001.

[7] A.Borgida, J.Mylopoulos, H.K.T.Wong; *Generalization/Specialization as a Basis for Software Specification*; in "On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages", 87-117, Springer Verlag, 1984.

[8] Bright, M. and Hurson, A. and Pakzad, S. (1994) Automated Resolution of Semantic Heterogeneity in Multidatabases. ACM Transactions on Database Systems, 19(2), 212-253.

[9] M.L.Brodie, J.Mylopoulos, J.W.Schmidt (Eds.); *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer-Verlag, 1984.

[10] L.Cardelli; *A Semantics of Multiple Inheritance*; Info.&Comp., 76, 138-164; 1988 (Preliminary version in LNCS 173, Springer-Verlag, 1984).

[11] S.Castano, V.De Antonellis, M.G.Fugini, B.Pernici; *Conceptual Schema Analysis: Techniques and Applications*; ACM Transactions on Database Systems, 23(3), 286-332, 1998.

[12] Collins, A. and Loftus, E. (1975) A Spreading Activiation Theory on Semantic Processing. Psychological Review, 82, 407-428.

[13] R.Costello; *XML Schema Tutorial*, http://www.w3.org/XML/Schema, 2002.

[14] E.Damiani, A.Formica, M.G.Fugini, M.Missikoff, R.Pizzicannella; *Reusing Analysis Schemas in ODB Applications: a Chart Based Approach*; First East-European Symposium on Advances in Databases and Information Systems, St.Petersburg, Russia, September 2-5, 406-415, Nevsky Dialect, Russia, 1997.

[15] Y.Ding, D.Fensel, M.Klein, B.Omelayenko; *The semantic web: yet another hip?* Data & Knowledge Engineering Vol.41, No.2-3, 205-227, 2002.

[16] A.Formica; *Satisfiability of Object-Oriented Database Constraints with Bag and Set Attributes*; Information Systems, Vol.28, No.3, 213-224, 2003.

[17] A.Formica, H.D.Groger, M.Missikoff; *Object-Oriented Database Schema Analysis and Inheritance Processing: a Graph-Theoretic Approach*; Data & Knowledge Engineering (DKE) Vol.24, No.2, 157-181, 1997.

[18] A.Formica, M.Missikoff; *Concept Similarity in SymOntos: an Enterprise Ontology Management Tool*; The Computer Journal, Vol.45, No.6, 583-594, 2002.

[19] Z.Galil; *Efficient algorithms for finding maximum matching in graphs*; ACM Computing Surveys, 18, 23-38, 1986.

[20] M.Jarke, M.Lenzerini, Y.Vassiliou; *Fundamentals of Data Warehouses*; Springer-Verlag, Berlin, 1999.

[21] D.Lee, W.W.Chu; *Comparative Analysis of Six XML Schema Languages*; SIGMOD Record 29(3), 76-87, 2000.

[22] M.Missikoff, X.F.Wang; *A group decision system for collaborative ontology building*; Proceedings of Int'l Conference on Group Decision and Negociation, La Rochelle, France, June 4-7, 153-160, 2001.

[23] Rada, R. and Mili, H. and Bicknell, E. and Blettner, M. (1989) *Development and application of a metric on semantic nets*; IEEE Transactions on Systems, Man, and Cybernetics, 19(1), 17-30.

[24] N.Routledge, L.Bird, A.Goodchild; *UML and XML Schema*; Australasian Database Conference (ADC), Melbourne, Australia, 2002.

[25] M.Uschold, M.Gruninger; *Ontologies: Principles, Methods and Applications*; The Knowledge Engineering Review, V.11, N.2, 1996.

[26] The World Wide Web Consortium (W3C); http://www.w3.org/XML/Schema.

[27] R.Xiaou, T.S.Dillon, E.Chang, L.Feng; *Modeling and Transformation of Object-Oriented Conceptual Models into XML Schema*; Database and Expert Systems Applications (DEXA), Munich, Germany, 795-804, 2001.