ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

A. Formica

SIMILARITY OF XML-SCHEMA ELEMENTS: A STRUCTURAL AND INFORMATION CONTENT APPROACH

R. 656 Gennaio 2007

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 The Computer Journal 51(2), pp.240-254, 2008.

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 are the emerging standards for describing and validating semi-structured documents across the Internet, due to the rich set of modeling constructors, types, and constraints they provide. Semantic similarity is growing in importance in different settings, such as digital libraries, heterogeneous databases and, in particular, the Semantic Web. The focus of this paper is the definition of a method for determining semantic similarity of XML-Schema elements in the presence of *type hierarchies*. Such a method has been defined by combining and revisiting: (i) the *information content* approach, and (ii) a method for comparing the structural components of type declarations, inspired by the *maximum weighted matching* problem in bipartite graphs.

Keywords: *Semantic Web, XML-Schemas, type hierarchies, information content, similarity reasoning.*

1. Introduction

XML(eXtensible Markup Language)-Schemas are the emerging standards for describing and validating semi-structured documents across the Internet, due to the rich set of modeling constructors, types, and constraints they provide [1]. Semantic similarity is growing in importance in different settings, such as digital libraries, heterogeneous databases and, in particular, the Semantic Web [2, 3, 4, 5]. In fact, with the increasing number of information on the Web and the proliferation of Web services, advanced methods and tools are required for enabling semantic Web service discovery, Web data extraction, inter-enterprise electronic commerce interactions, etc.. For instance, enterprises describe their own Web services, and semantic Web service discovery consists in the identification of existing Web services that can potentially be used within new Web applications [6]. Current technologies enable enterprises to discover and determine how to interact with services fronted by other business, however they do not address the problem of how to assess the similarity of - and, possibly, to reconcile - heterogeneous service descriptions [7].

Similarity measures for XML-Schema elements can support semi-automatic and labor-intensive activities, such as XML-Schema integration, XML-Schema matching [8], and the critical step of (XML-based) Web service discovery [6]. Furthermore, they can support existing proposals for determining Web service similarity, as for instance [9], which provides a suite of methods to assess the similarity between WSDL (Web Service Description Language) specifications, inspired by traditional information retrieval techniques.

The aim of this paper is therefore the definition of a method for determining semantic similarity of XML-Schema elements, which can (partially or totally) support the critical activities mentioned above. Note that this work focuses on the similarity of concepts at schema level

(metadata), rather than instances (data). Furthermore, concepts are expressed according to a subset of the *XML-Schema specification* defined by *W3C* [1]. In particular, an important modeling notion of XML-Schemas has been addressed: XML-Schema *type hierarchies*. With this regard, in the literature the natural, time-honored way to determine semantic similarity in a taxonomy is based on the so-called *edge-counting* approach [10, 11] - that is, the shorter the path between nodes, the more similar the concepts associated with the nodes. Unfortunately, this approach relies on the assumption that links in the taxonomy represent uniform distances that, in general, is a characteristic very difficult to find in real taxonomies. For this reason, a different approach has been proposed in the literature based on the notion of *information content* [12, 13], which is independent of the path lengths of the hierarchy. Such an approach has been refined in [5], where a similarity measure showing a higher correlation with human judgments has been defined [14].

The semantic similarity method proposed in this paper has been defined by combining and revisiting: (i) the *information content* approach, as defined in [5], regarding hierarchically related concepts, and (ii) a *structural* approach for comparing sets of attributes and sequences of elements inspired by the *maximum weighted matching* problem in bipartite graphs [15], regarding type declarations. The method mentioned at point (ii) is also at the basis of the similarity reasoner *SymOntos*, an ontology management system presented in [16]. In this paper, as better clarified in the next section, the *SymOntos* approach has been revisited and extended by considering: (a) the information contents of type declaration components, rather than the axiomatic similarity degrees provided by panels of experts in the application domains, (b) the similarity of sets of attributes and sequences of elements, as required by XML-schemas, (c) the similarity of the types associated with attributes and elements within type declarations. This extension allows us to overcome some of the limitations of other structural approaches defined in the literature.

In this paper, we will see that the radical difference between the information content and the structural approaches often leads to incomparable concept similarity scores. For this reason, the contribution of this paper, besides the extension of the *SymOntos* method, consists in the formalization of a similarity measure which combines and reduces the distances between these approaches.

The paper is organized as follows. In the next section, the related work is given, and in Section 3 the XML-Schema data model addressed in this paper is informally presented via examples. Successively, in Section 4, the basic definitions are provided in formal terms. In particular, a first contribution of this paper is the formalization of a subset of the W3C XML Schema specification and, successively, the related notion of *expanded form*. In Section 5, the similarity method is presented. In particular, Subsection 5.1 deals with point (i) above, and introduces the *information content similarity* (*ics*), whereas Subsection 5.2 addresses point (ii) above, and presents the notion of *type structural similarity* (*tss*), whose formalization is a further contribution of this paper. Such notions are then integrated and, in the last subsection, the definition of *combined similarity* (*CombSim*) for elements is given. Then, Section 6 is presented, where an illustrative data set is shown comparing the proposed approach with some among the most widespread approaches in the literature. Successively, the Conclusion and Future Work Section follows.

2. Related Work

In the literature many similarity measures have been proposed, depending on the research fields, the data models, or specific applications [5, 4]. In the following we will focus on the existing contributions dealing with hierarchically related concepts and numerical feature vectors (in our case, XML-Schema sequences of elements and sets of attributes). With this regard, it is inter-

esting to recall the approach adopted in [17] in the context of data integration, or in [6] within XML-based Web service discovery. In the former paper, when dealing with hierarchically related concepts, similarity is determined by associating a constant value with *any* pair of hierarchically related concepts. In the latter paper the authors address the hierarchical organization of concepts by assigning a similarity coefficient to nouns, referred to as *Name Affinity coefficient*, which allows the determination of the *Entity-based similarity coefficient* between service descriptions. Note that the Name Affinity coefficient is defined by simply assigning a constant value, depending on the levels of specializations of the concepts in the ISA hierarchy. Regarding the numerical feature vectors, in both the mentioned papers the *Dice*'s function is adopted [18], that is formally recalled in Section 6. In particular, with respect to the *type structural similarity* (*tss*) proposed in this paper, the *Dice*'s function allows concept similarity to be computed without explicitly considering the similarity degrees of components. Furthermore, in *Dice* the types associated with features are not addressed in the comparison. Analogously in [19], within *Semantic Networks*, semantic relatedness (similarity) 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. As opposed to this approach, in this paper similarity is computed by combining the *information content similarity* (*ics*), regarding hierarchically related concepts, and the *tss*, regarding features vectors. In addition, the *tss* is computed according to the *ics* of components.

A well-known method to measure similarity within a taxonomy is referred to as *edge-counting* approach. In particular, in [11] the *conceptual distance* has been defined by considering the length of the shortest path connecting the concepts in the taxonomy. Analogously, in [20] the *Semantic-Distance Metric* has been defined, consisting of a weighted count of the links of the paths connecting concepts, where not only hyperonym/hyponym links are addressed, but also synonym links. But, as already mentioned in the Introduction, the widely acknowledged problem

with this approach is that it assumes that links in the taxonomy represent uniform distances that, in general, is not true in real world taxonomies. For this reason, in this paper the *information content* approach proposed by Resnik [12], and successively refined by Lin [5], has been followed, which is independent of path lengths. A further approach that, under specific assumptions, can be seen as a special case of that proposed by Lin has been defined by Wu&Palmer in [21]. A comparison among the proposals of Resnik, Lin, and Wu&Palmer (that have been formally recalled in this paper) is given in Section 6.

Within Artificial Intelligence, it is worth recalling [22], and [23], where both the intensional and extensional levels of concepts have been addressed. However, in the former paper, there are a number of limitations, such as the necessity that two concepts are at the same hierarchy level to be compared. In the latter paper, general forms of distance metrics have been defined, although with more emphasis on the similarity between instances, rather than concepts.

Similarity of XML-Schema elements has been addressed by the author in [24], by following a different approach with respect to the one proposed in this paper. In fact in the mentioned work, similarity is computed by making use of domain *ontologies* [2, 3]. In particular, an *axiomatic similarity* degree is associated with each pair of element and attribute names of the XML-Schema, which is established by a panel of experts in the given domain by means of a *Consensus System* [25]. Therefore, the main drawback of the proposal presented in [24] consists in relying on human domain expertise. As opposed to that approach, here in place of the axiomatically given similarity, the *ics* has been addressed, in line with the probabilistic approach for determining similarity within ISA taxonomies defined in [12, 5].

In [16] the similarity reasoner underlying the ontology management system *SymOntos* has been presented. In particular, in *SymOntos* concept similarity is determined according to a notion, namely the *flat structural similarity* (*fss*), which is a special case of the *tss* presented in

this paper. In fact, in *SymOntos* attributes are not typed, sequences of elements are not defined and, as in the case of [24], the comparison is based on the ontology-based notion of axiomatic similarity degree provided by a panel of experts in the application domain. In this paper, the replacement of the *fss* with the *tss* allows us to refine similarity scores in the case of attributes (or elements) with the same names and different types (with this regard, see Section 6). In addition, the replacement of the axiomatic similarity degree with the *ics* degree allows us to automatically compute element and attribute similarity by relying on any lexical database for the English language (such as, for instance, *WordNet* [26]), therefore without the need of human expertise. Finally, it is worth recalling that in *SymOntos* a specific notion for hierarchically related concepts has been introduced, namely the *hierarchical structural similarity* (*hss*). Such a notion is based on the *extensional* aspect of inheritance, i.e., on the distribution of concept instances along the ISA hierarchy. In particular, it is related to the degree of refinement between concepts: the greater the refinement, the higher the distance between the concepts. In this paper, the notion of *hss* defined in *SymOntos* has been replaced with the probabilistic approach defined in [12], which is based on the information content shared by concepts within the ISA hierarchy.

It is worth mentioning that the need of defining a similarity measure that overcomes the limitations of the approaches proposed in the literature is also supported by other authors, as for instance in [4]. Although this work falls in the area of Geographic Information Systems, in that paper the authors ignore the geometric aspects of similarity and concentrate on the cognitive properties of semantic similarity. They state that the similarity models defined in the literature are often in contrast and, in particular, they refer to the work done by psychologists and computer scientists. The formers focus on the features (properties or descriptors) of concepts, whereas the latters typically address the interrelations of concepts organized according to a hierarchy. For this reason in [4] a method aimed at combining both these approaches is pro-

posed, referred to as *Matching-Distance Similarity Measure* (MDSM). It is defined by extending the feature-matching model of Tversky with a semantic distance method based on the traditional edge-counting approach. These authors also argue that the information content approach represents an interesting research topic, but further proposals have not yet been done in that direction[1].

It is worth recalling that in [27], the *SOQA-SimPack Toolkit* has been presented, aiming at evaluating similarities within a given ontology, and between concepts of different ontologies. In particular, it provides a generic and extensible library of ontological similarity measures in order to capture various notions of similarity.

Finally, we recall that in the literature, a lot of work has been developed regarding the modeling capabilities and the expressive power of XML-Schemas. For instance, see [28] where DTDs and XML Schema definitions have been compared, [29] where six noteworthy XML-Schema languages have been analyzed, [30] where a systematic approach to the data modeling capabilities of XML-schemas has been given, [31] where a formal framework supporting the legality of XML-Schema type hierarchies has been proposed, or [32] where an algorithm for detecting changes between versions of XML documents has been defined. However, all the mentioned papers do not address any similarity measurements.

[1]Note that MDSM has not been addressed here since in [4] the authors state that a higher correlation between their proposal and human answers depends on the accurate identification of distinguishing features of the geographic classes. In the case the identification of class attributes is not accurate, the experimental results show that the information content approach has a higher correlation with human judgement than MDSM.

10.

3. The XML-Schema Data Model

In this section the XML-Schema data model addressed in this paper is informally presented. It focuses on a subset of the *W3C XML-Schema specification* defined in [1].

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*). Among the simpleTypes defined in XML-Schemas, this paper focuses on the atomic types *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. Note that the *choice* and *all* primitives are not included in the data model addressed by this paper, and concern future work.

SimpleTypes and 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 [1], a type can be defined in terms of one base type at most, that is, by using the Object-Oriented terminology [33], in an XML-Schema multiple inheritance is not allowed. In this paper, derived simpleTypes (e.g., restrictions of atomic types by enumeration or by specifying the interval extremes) are not addressed, and we focus only on type hierarchies involving complexTypes.

The *extension base* constructor allows the incremental definition of complexTypes by adding elements and/or attributes to existing complexTypes. The derived complexType will have all the elements and attributes of the base type, plus the additional elements and/or attributes declared for it. Notationally, the *complexContent* constructor is used, as shown in the example below.

Example 3.1. Consider the following declarations:

```
<xsd:element name="person" type="personType"/>
<xsd:element name="worker" type="workerType"/>
<xsd:element name="activity" type="activityType"/>

<xsd:complexType name="personType">
   <xsd:sequence>
      <xsd:element name="firstName" type="xsd:string"
            minOccurs ="1" maxOccurs="2"/>
      <xsd:element name="lastName" type="xsd:string"/>
   </xsd:sequence>
   <xsd:attribute name="age" type="xsd:integer"/>
</xsd:complexType>

<xsd:complexType name="workerType">
   <xsd:complexContent>
      <xsd:extension base="personType">
         <xsd:sequence>
            <xsd:element name="occupation" type="activityType"
                  minOccurs="1" maxOccurs="2"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="activityType">
   <xsd:attribute name="name" type="xsd:string"/>
   <xsd:attribute name="dailyHours" type="xsd:integer"/>
</xsd:complexType>
```

In this example, three element declarations are present, with names *person*, *worker*, and *activity*, and types *personType*, *workerType*, and *activityType*, respectively. For instance, in the case of *personType*, two elements are declared, namely *firstName* and *lastName*, both of type *string*, and one attribute, namely *age*, of type *integer*. The complexType *personType* is the base type of the derived type *workerType*. In particular, the latter is defined by the additional element *occupation* of type *activityType*. □

Note that it is possible to *inline* unnamed complexTypes in element declarations [34]. Since

elements which are declared by inlining unnamed types can always be transformed into flat form by introducing type names, in this paper, for the sake of simplicity, we will focus on named complexTypes. Furthermore, according to the XML model [1, 34], in a complexType attributes with the same names are not allowed, whereas elements with the same names are allowed, providing they have the same types.

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 replacing the types of one or more elements/attributes of the base type with derived types, therefore reducing the ranges of values that elements/attributes can take. The notation is similar to the one used for the extension base constructor.

Example 3.2. Consider the following types:

```
<xsd:complexType name="employeeType">
   <xsd:complexContent>
      <xsd:restriction base="workerType">
         <xsd:sequence>
            <xsd:element name="occupation" type="jobType"
               minOccurs ="1" maxOccurs="2"/>
         </xsd:sequence>
      </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="jobType">
   <xsd:complexContent>
      <xsd:extension base="activityType">
         <xsd:attribute name="pay" type="xsd:integer"/>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

In this case *workerType*, as defined in the previous example, is the base type of *employeeType*. In particular, *employeeType* has been derived by replacing the type associated with *occupation* in the base type, i.e., *activityType*, with *jobType*. (Such a restriction is legal since *activityType*

is the base type of *jobType*. For further details, refer to [31].)

□

Note that in XML-Schemas recursive complexTypes (i.e., defined by types that, directly or indirectly, refer to the type being defined) are not allowed. Furthermore, it is possible to specify the minimum number (*minOccurs*) and maximum number (*maxOccurs*) of occurrences of a given element where, in general, the maximum number can also be *unbounded*. (This does not hold in the case of attributes for which only single occurrences are allowed.) For instance, in the case of *personType*, the element *firstName* may have at least one and at most two occurrences. Note that in this paper, we focus on bounded *maxOccurs*. An extension of the similarity method to unbounded *maxOccurs* is feasible, but concerns future work. In the next section, the XML-Schema data model addressed in this paper is formally defined.

4. Basic Definitions

In the following $\mathcal{T}$, $\mathcal{E}$, and $\mathcal{A}$ are countable sets of type names, element names and attribute names, respectively. In this section, for the sake of simplicity, in place of the XML-Schema syntax introduced above, a more compact syntax will be used. Let us start with the definition of a *type*.

Definition 4.1. [Type declaration]. A *type declaration* (*type* for short) (of name) τ is defined as follows:

$$\tau = \prec tp \succ \mid \mathbf{ext}(\sigma) \prec tp \succ \mid \mathbf{res}(\sigma) \prec tp \succ$$

$$\prec tp \succ ::= < e :\prec \gamma \succ_{m,M} [, e :\prec \gamma \succ_{m,M}] > \mid \{ a :\prec \beta \succ [, a :\prec \beta \succ] \}$$

$$\mid < e :\prec \gamma \succ_{m,M} [, e :\prec \gamma \succ_{m,M}] > \{ a :\prec \beta \succ [, a :\prec \beta \succ] \}$$

$$\prec \gamma \succ ::= \lambda \mid \prec \beta \succ$$

14.

$\prec \beta \succ$ ::= **string** | **integer** | **decimal** | **boolean** | **date** | **time**

where τ, σ, $\lambda \in \mathcal{T}$, $e \in \mathcal{E}$, and $a \in \mathcal{A}$. Angular parentheses (<,>) stand for sequence, curly braces ({,}) stand for set, and square brackets ([,]) stand for iteration (zero, one or more). Furthermore, m and M denote natural numbers standing for the minimum (*minOccurs*) and the maximum (*maxOccurs*) numbers of occurrences of the element e, respectively. In the absence of occurrence indicators, we assume, by default, m,M=1. Atomic types, that will be denoted by the set $\mathcal{B}$ = {**string, integer, decimal, boolean, date, time**}, are also referred to as *simpleTypes*, whereas type declarations are referred to as *complexTypes*. In the case of the *extension base* (**ext**) or *restriction base* (**res**) constructs, the type σ is referred to as the *base type* of τ, and τ is a *derived type*. Recursive types are not allowed. Note that $\mathcal{T} \cap \mathcal{B} = \emptyset$. □

The notion of a *flat element* follows, taking into account that, as mentioned previously, elements which are not flat can always be transformed into flat forms [34, 1].

Definition 4.2. [Element declaration]. A *flat element declaration* (*element* for short) (of name) $e \in \mathcal{E}$ is defined as follows:

e:τ

where $\tau \in \mathcal{T} \cup \mathcal{B}$. □

Example 4.1. According to the compact syntax above, for instance, the element *worker* of Example 3.1 becomes:

`worker`:`workerType`

and *workerType*:

`workerType` = **ext**(`personType`) <`occupation:activityType`$_{1,2}$>

whereas *personType* can be rewritten as:

`personType` = <`firstName:string`$_{1,2}$,`lastName:string`> {`age:integer`}

□

Below the notion of an XML-*Schema* is introduced. Note that, according to W3C, this notion provides a structure- and data type definition for XML data, expressing a subset of the W3C XML Schema specification. In this paper, for the sake of simplicity, we refer to it as an XML-Schema.

Definition 4.3. [**XML-Schema**]. An XML-*Schema* Σ is a 5-tuple:

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

where $T \subset \mathcal{T}$ (T is a set of type names), $E \subset \mathcal{E}$ (E is a set of element names), $A \subset \mathcal{A}$ (A is a set of attribute names), and T_{dec}, E_{dec} are sets of declarations of types and elements (according to Definitions 4.1 and 4.2), respectively, with names in T, E, A, such that the following holds:

- every type name in T has exactly one type declaration in T_{dec};
- every type name in T_{dec} and E_{dec} is in T;
- let $\mathcal{H}_\Sigma$ be a directed tree associated with Σ, defined as follows. The root of the tree is a type, named *anyType*, which is not associated with any element or attribute (it does not have any declaration). Let T^+ be the set $T \cup \{anyType\}$ and *dirDer* (*directDerivation*) be the ordered binary relation on T^+ defined as follows:
  - *dirDer*(*anyType*,τ), for any type $\tau \in T$ which is not a derived type;
  - *dirDer*(σ, τ), if $\tau \in T$ is a derived type and $\sigma \in T$ is the base type of τ.

  Then $\mathcal{H}_\Sigma = (T^+, dirDer)$ is referred to as the *type hierarchy* of Σ (of root type *anyType*).

□

Note that the type hierarchy $\mathcal{H}_\Sigma$ has been defined in line with the notion of a *signature for inheritance* originally introduced in [35] and successively refined in [36, 31]. In particular, cycles

in $\mathcal{H}_\Sigma$ are not allowed since derived types cannot be used (as base types) to define their own base types.

Example 4.2. Consider the following sets of element and type declarations, which include the elements and types of Examples 3.1, 3.2. The XML-Schema defined starting from these sets of declarations will be the running example of this paper.

E_{dec} = {
person:personType
student:studentType
collegian:collegianType
worker:workerType
employee:employeeType
jobHolder:jobHolderType
lover:loverType
boyFriend:boyFriendType
girlFriend:girlFriendType
activity:activityType
job:jobType
}

T_{dec} = {
studentType = **ext**(personType) {matriculation:integer}
collegianType = **ext**(studentType) {college:string}
jobHolderType = **ext**(employeeType) {insurance:string,salary:integer}
loverType = **ext**(personType) {lovedPerson:string}
boyFriendType = **ext**(loverType) {indemnity:string,earnings:integer}
girlFriendType = **ext**(loverType) {educationInstitute:string}
....}

where personType, workerType, employeeType, activityType, and jobType are given in Examples 3.1, 3.2. The type hierarchy of the schema is represented in Figure 1.

□

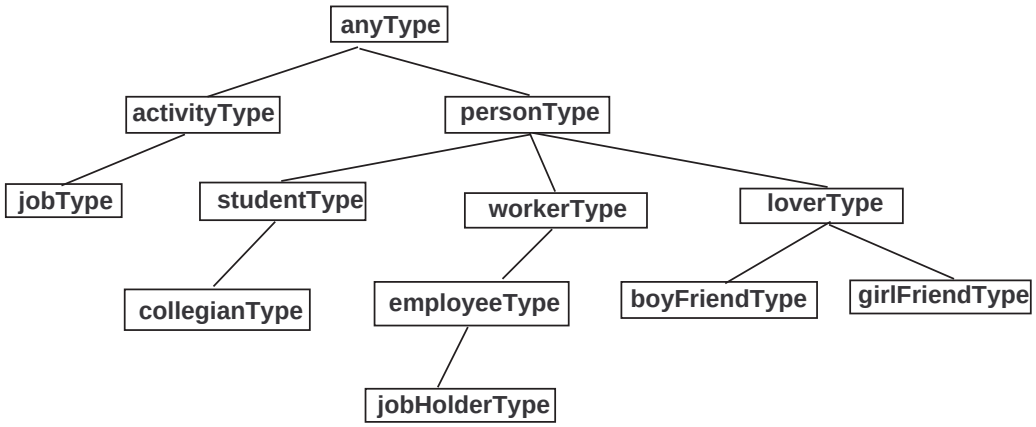

Figure 1: Type Hierarchy of Example 4.2

4.1. Type Expansion

A preliminary step for determining element similarity according to the proposed method is related to the *expansion* of complexTypes. Type expansion concerns the well-known problem of *inheritance*, widely investigated in the literature, see for instance [37, 38, 36]. In essence, it consists in the replacement of the generalization/specialization constructor of types (in this case, the *extension* and *restriction base*) with the properties (elements, with their occurrence indicators, and attributes) of the related supertypes (base types) along the hierarchy (ancestors). This is a necessary step for computing similarity, since all the elements and attributes of types, including that of all their ancestors, have to be compared. Type expansion in XML-Schemas has been extensively investigated in [31]. In particular, in the mentioned paper an algorithm for the removal of the *extension* and *restriction base* constructors has been presented that provides the *expanded form* of an XML-Schema. Note that a prerequisite for the successful termination of the algorithm is the absence, in the schema, of *illegal* type declarations (that can arise, for instance, by extending types with attribute names defined in the base types). In this paper, we assume that XML-Schema type hierarchies are legal, therefore with complexTypes whose expansions

are defined (the problem of the legality of XML-Schema type hierarchies goes beyond the scope of this paper).

Definition 4.4. [Expanded form of an XML-Schema] The *expanded form* of an XML-Schema $\Sigma = (T, E, A, T_{dec}, E_{dec})$ is a pair $\mathcal{S} = (\Sigma', \mathcal{H}_\Sigma)$, where Σ' is the schema Σ where each derived complexType τ has been replaced with its expansion τ', and $\mathcal{H}_\Sigma$ is the type hierarchy of Σ. □

Note that in the expanded form of an XML-Schema the information concerning base types, which is lost once type declarations have been expanded, is contained in the type hierarchy $\mathcal{H}_\Sigma$.

Example 4.3. Below, the expansions of some of the type declarations of Example 4.2 are given.

workerType = <firstName:string$_{1,2}$,lastName:string,occupation:activityType$_{1,2}$>
{age:integer}
employeeType = <firstName:string$_{1,2}$,lastName:string,occupation:jobType$_{1,2}$>
{age:integer}
jobHolderType = <firstName:string$_{1,2}$,lastName:string,occupation:jobType$_{1,2}$>
{age:integer,insurance:string,salary:integer}
boyFriendType = <firstName:string$_{1,2}$,lastName:string>
{age:integer,lovedPerson:string,indemnity:string,earnings:integer}
girlFriendType = <firstName:string$_{1,2}$,lastName:string>
{age:integer,lovedPerson:string,educationInstitute:string}

□

4.2. Weighted type hierarchy

The notion of a *weighted type hierarchy* is the basis of the element similarity method proposed in this paper. It consists in the association of weights with the types of the hierarchy, standing for the *probabilities* that randomly selected instances are of that types. Such a notion is in line with [12, 13], and in this work has been proposed within XML-Schema type hierarchies.

Definition 4.5. [Weighted type hierarchy of an XML-Schema] Given an XML-Schema $\Sigma = (T, E, A, T_{dec}, E_{dec})$, its expanded form $\mathcal{S} = (\Sigma', \mathcal{H}_\Sigma)$, consider the type hierarchy $\mathcal{H}_\Sigma$

augmented with a function p: $T^+ \rightarrow [0,1]$ s.t. for each type $\tau \in T$, $p(\tau)$ is the *probability* that any instance is of type τ, and $p(anyType) = 1$. Then the type hierarchy is referred to as *a weighted type hierarchy* of Σ, and it will be denoted by $\mathcal{H}^p_\Sigma$.

□

In the literature, probabilities are estimated according to the *frequencies* of concepts (types). In particular, given a concept τ, the *probability* $p(\tau)$ is defined as follows:

$p(\tau) = \frac{freq(\tau)}{M}$

where $freq(\tau)$ is the *frequency* of the concept τ estimated using noun frequencies from large text corpora, as for instance the Brown Corpus of American English [39], and M is the total number of observed instances of nouns in the corpus. Note that each concept's noun that occurs in the corpus is also counted as an occurrence of each more abstract concept up in the hierarchy. For instance, in Figure 1, an occurrence of *employee* is also counted as an occurrence of *worker* and *person* (for further details, see [12]).

In the following, the expanded form of an XML-Schema containing a weighted type hierarchy will be denoted by $\mathcal{S}^p = (\Sigma', \mathcal{H}^p_\Sigma)$ and it will be referred to as a *weighted expanded form* of an XML-Schema.

In this paper probabilities have been defined according to *SemCor* project [40] which labels subsections of Brown Corpus to senses in the *WordNet* lexicon [26], but any other lexical database for the English language can be used as well. According to *SemCor*, the total number of observed instances of nouns in the corpus is $88,312$.

In Table 1 the frequencies of the concepts of our running example are given. Note that missing frequencies in *WordNet*, as for instance in the case of *boyFriendType* or *jobHolderType*, have been assumed equal to 1.

The weighted type hierarchy of Example 4.2 is given in Figure 2, where the numbers associated with the nodes of the hierarchy are the probabilities of the concepts.

Table 1: Concept frequencies and definitions according to *WordNet*

frequency	**concept**	**definition**
7229	person	*a human being*
68	student	*a learner who is enrolled in an educational institution*
1	collegian	*a student (or former student) at a college or university*
289	worker	*a person who works at a specific occupation*
57	employee	*a worker who is hired to perform a job*
1	jobholder	*an employee who has a regular job*
7	lover	*a person who loves or is loved*
1	boyfriend	*a man who is the lover of a girl or young woman*
1	girlfriend	*a girl or young woman with whom a man is romantically involved*
614	activity	*any specific activity*
280	job	*a specific piece of work required to be done as a duty or for a specific fee*

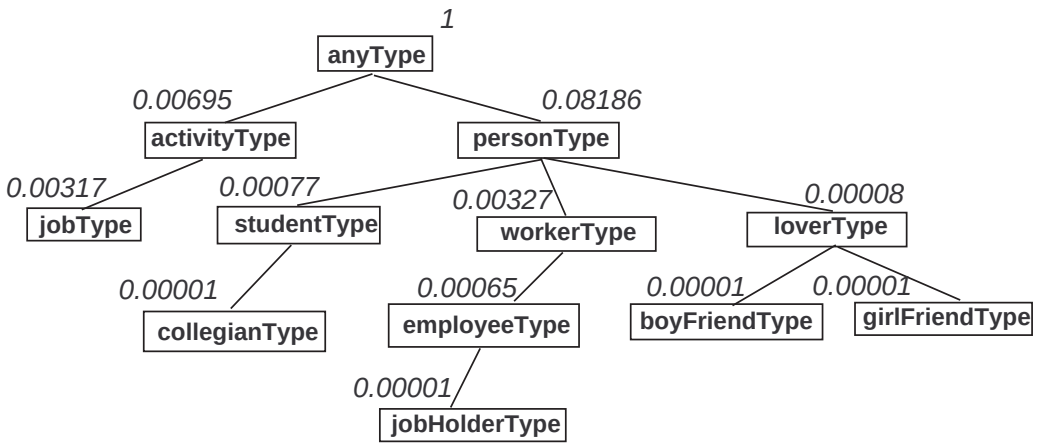

Figure 2: Weighted Type Hierarchy of Example 4.2

Before concluding this section, the notion of a *SynSet* for an XML-Schema is introduced. In the next section we will see how this notion plays a fundamental role in determining XML-Schema element similarity.

Definition 4.6. **[SynSet for an XML-Schema]** Given an XML-Schema $\Sigma = (T, E, A, T_{dec}, E_{dec})$, let $B_i \subset \mathcal{T} \cup \mathcal{E} \cup \mathcal{A}$, $i = 1...n$, be sets of synonyms defined according to a lexical database for the English language such that, for each i, $B_i \cap (T \cup E \cup A) \neq \emptyset$. Then, the (possibly empty) set:

$SynSet_\Sigma = \{B_1, ..., B_n\}$, $n \geq 0$,

is referred to as a *set of synonyms* for the XML-Schema Σ. □

Example 4.4. For instance, according to *WordNet*, the following is a set of synonyms for the XML-Schema Σ of our running Example:

$SynSet_\Sigma = \{$

$\{educationInstitute, college\}$

$\{insurance, indemnity\}$

$\{pay, wage, salary, earnings\}$

$\}$

□

5. Similarity of XML-Schema Elements

In this section the method for determining similarity of XML-Schema elements is introduced. It is based on the combination of two notions: (i) the *information content similarity*, devoted to measure the similarity of type, element, and attribute names, on the basis of the type hierarchy and the set of synonyms associated with the schema, and (ii) the *type structural similarity*, conceived to determine the similarity of sequences of typed elements and sets of typed attributes. In the following subsections the two notions are presented and, finally, in Subsection 5.3, the overall method is introduced by combing them.

5.1. Information Content Similarity

The notion of *information content similarity* is based on the definition of *semantic similarity* in a taxonomy, as introduced in [12, 13]. In particular, such a definition follows the standard argumentation of information theory [41], for which the *information content* of a concept n is defined as:

$$-\log p(n)$$

that is, as the probability of a concept increases, the informativeness decreases, therefore the more abstract a concept, the lower its information content. In particular, following Resnik's approach, the semantic similarity (Sim_R) of two concepts n_1,n_2, organized according to a taxonomy (whose root represents the most abstract concept) is given by the maximum information content shared by the concepts, that is:

$$Sim_R(n_1, n_2) = \max_{n \in \mathcal{U}(n_1,n_2)} [-\log p(n)]$$

where $\mathcal{U}(n_1, n_2)$ is the set of concepts that are upper bounds of (informally, the set of concepts that are more abstract than) both n_1,n_2 in the hierarchy. In other words, the more information

two concepts share, the more similar they are. Of course, if the *least upper bound* (*lub*) (the less abstract concept among the upper bounds) of n_1,n_2 is defined in the taxonomy, it provides the maximum information content shared by n_1,n_2.

Note that, since in the case of XML-Schemas the *lub* of two types is always defined and unique, the notion of Sim_R can be simplified as follows:

$$Sim_R(n_1, n_2) = -\log p(lub(n_1, n_2))$$

In the following, the notion of *information content similarity* is formally introduced. In particular, it is inspired by the approach proposed by Lin in [5], that is very similar to that of Resnik. Essentially, with respect to the latter, besides the information content provided by the *lub*, in [5] also the information contents of the comparing concepts are addressed, as shown below.

Definition 5.1. [Information content similarity (ics)] Given an XML-Schema $\Sigma = (T, E, A, T_{dec}, E_{dec})$, a weighted expanded form $\mathcal{S}^p = (\Sigma', \mathcal{H}^p_\Sigma)$ of Σ, and a $SynSet_\Sigma = \{B_1, ..., B_h\}$, $h \geq 0$ for Σ, consider n_1,$n_2 \in T \cup E \cup A \cup \mathcal{B}$. Then, the *information content similarity* of n_1,n_2, denoted by $ics(n_1, n_2)$, is defined as follows:

1. if $n_1 = n_2$, or n_1,$n_2 \in B_k \in SynSet_\Sigma$, for some k, $1 \leq k \leq h$: $ics(n_1, n_2) = 1$;

2. if $n_1 \neq n_2$, n_1,$n_2 \in T$ and n_1,$n_2 \notin B_k \in SynSet_\Sigma$, for any k:

$$ics(n_1, n_2) = \frac{2 \log p(lub(n_1, n_2))}{\log p(n_1) + \log p(n_2)}$$

3. $ics(n_1, n_2) = 0$ otherwise.

□

Note that points 1. and 3. apply in the cases n_1, n_2 are element, attribute, atomic type, or complexType names, whereas point 2. is restricted to complexType names.

Example 5.1. Consider *studentType* and *jobHolderType* of our running example. The maximum information content shared by these concepts is provided by $personType$, that is their *lub* according to the hierarchy of Figure 1. Therefore, their *ics* is the following:

$ics(studentType, jobHolderType) = \frac{2 \log p(personType)}{\log p(studentType) + \log p(jobHolderType)} = \frac{2*3.61}{10.34+16.43} = 0.27$

whereas, if we consider *employeeType* and *jobHolderType*, that are directly related in the hierarchy, the following holds:

$ics(employeeType, jobHolderType) = \frac{2 \log p(employeeType)}{\log p(employeeType) + \log p(jobHolderType)} = \frac{2*10.60}{10.60+16.43} = 0.78.$

□

As mentioned above, in the case of $Sim_R(n_1, n_2)$ concept similarity depends only on the shared maximum information content (i.e., in XML-Schemas, the information content of the *lub*), whereas, according to the definition above, it also depends on the comparing concepts. For instance, according to $Sim_R(n_1, n_2)$, the similarity scores of the pairs ($student$,$jobHolder$), ($collegian$,$boyFriend$), and ($collegian$,$girlFriend$) coincide since they all have the same *lub*, represented by $person$. This point is also discussed in Section 6.

5.2. Type Structural Similarity

The notion of *type structural similarity* presented in this subsection is an extension of the notion presented in [24], originally introduced in [16]. It is defined for expanded type declarations and can be led back to the *maximum weighted matching* problem in bipartite graphs, that can be solved in polynomial time [15]. In order to present it, the following notions are introduced.

Definition 5.2. [$elemOf_{min}$, $elemOf_{Max}$, *attrOf*, *typeOf*] Consider a type τ of the expanded form $\mathcal{S} = (\Sigma', \mathcal{H}_\Sigma)$ of an XML-Schema Σ, and suppose that:

$\tau = < e_i : \gamma_{i\ m_i,M_i} > \{a_j : \lambda_j\}$, $i = 1...k$, $j = 1...h$.

Then, $elemOf_{min}(\tau)$, $elemOf_{Max}(\tau)$, and $attrOf(\tau)$, are defined as follows:

$elemOf_{min}(\tau) = \{e_i\}^*$, $i = 1...k$, where for each i, the multiplicity of e_i is m_i,

$elemOf_{Max}(\tau) = \{e_i\}^*$, $i = 1...k$, where for each i, the multiplicity of e_i is M_i,

$attrOf(\tau) = \{a_j\}$, $j = 1...h$,

where the symbol $*$ means that $\{e_i\}^*$ is a bag (or a multiset).

Furthermore, consider an element e_i, $1 \leq i \leq k$, and an attribute a_j, $1 \leq j \leq h$, of τ. Then, $typeOf(\tau, e_i)$, and $typeOf(\tau, a_j)$ are defined as follows:

$typeOf(\tau, e_i) = \gamma_i$, where γ_i is the type of e_i declared in τ;

$typeOf(\tau, a_j) = \lambda_j$, where λ_j is the type of a_j declared in τ.

□

We recall that in the above definition, the bag (or multiset) symbol [42] has been introduced to deal with multiple element occurrences (whereas this does not hold for attributes).

Example 5.2. Consider *jobHolderType* (in particular the expanded form given in Example 4.3). Then:

$elemOf_{min}(jobHolderType) = \{firstName, lastName, occupation\}$

$elemOf_{Max}(jobHolderType) = \{firstName, firstName, lastName, occupation, occupation\}$

$attrOf(jobHolderType) = \{age, insurance, salary\}$

$typeOf(jobHolderType, firstName) = string$

$typeOf(jobHolderType, age) = integer.$

□

The notion of type structural similarity, which is inspired by the *maximum weighted matching* problem in bipartite graphs, can be informally explained as follows. Consider two complexTypes of names τ_i,τ_j, and one of $\{elemOf_{min}, elemOf_{Max}, attrOf\}$, for instance, *attrOf* (the following holds also for bags in the cases of $elemOf_{min}$, and $elemOf_{Max}$). Then:

- 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 attribute name. 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) [15];

- for each candidate set of pairs, consider the sum of the *ics* of the pairs. Then, one of the candidate sets having the maximum among all the computed sums is chosen.

Once the candidate set and bag of pairs with the maximum sums have been chosen for $attrOf$, $elemOf_{min}$, and $elemOf_{Max}$ respectively, type structural similarity takes also into account the *ics* of type declaration components. In formal terms, we start by introducing the notion of *set of candidate sets (bags) of pairs*.

Definition 5.3. [The set $\mathcal{C}_{\mathcal{R}}$ of candidate sets (bags) of pairs] Consider two complexTypes τ_i,τ_j of a weighted expanded form $\mathcal{S}^p = (\Sigma', \mathcal{H}^p_{\Sigma})$ of an XML-Schema Σ, and assume $\mathcal{R} \in \{elemOf_{min}, elemOf_{Max}, attrOf\}$. Let $n_{i,\mathcal{R}}$, $m_{j,\mathcal{R}}$ be the cardinalities of $\mathcal{R}(\tau_i)$, $\mathcal{R}(\tau_j)$, respectively, i.e., $n_{i,\mathcal{R}} = |\mathcal{R}(\tau_i)|$, $m_{j,\mathcal{R}} = |\mathcal{R}(\tau_j)|$, and suppose that $n_{i,\mathcal{R}} \leq m_{j,\mathcal{R}}$. Then, the *set* $\mathcal{C}_{\mathcal{R}}(\tau_i, \tau_j)$ *of candidate sets (bags) of pairs* is defined by all possible sets (bags) of $n_{i,\mathcal{R}}$ pairs of attribute (element) names as follows:

$$\mathcal{C}_{\mathcal{R}}(\tau_i, \tau_j) = \{\ \{\langle a_k, b_h\rangle_v\}^*_{v=1\ldots n_{i,\mathcal{R}}} \mid a_k \in \mathcal{R}(\tau_i),\ b_h \in \mathcal{R}(\tau_j), \text{ and } \forall \langle a_s, b_r\rangle_q,\ a_s \in \mathcal{R}(\tau_i),$$
$$b_r \in \mathcal{R}(\tau_j),\ 1 \leq q \leq n_{i,\mathcal{R}},\ (k = s \text{ or } h = r) \Rightarrow (a_k = a_s \text{ and } b_h = b_r)\}.$$

□

Example 5.3. For instance, assume that $\mathcal{R} = attrOf$ and consider the sets of attribute names of *studentType* and *jobHolderType* of Example 4.3, that are recalled below:

$attrOf(studentType) = \{age, matriculation\}$

$attrOf(jobHolderType) = \{age, insurance, salary\}$.

In this case, the following holds:

$\mathcal{C}_{attrOf}(studentType, jobHolderType) = \{$
$\{\langle age, age\rangle, \langle matriculation, insurance\rangle\},$
$\{\langle age, age\rangle, \langle matriculation, salary\rangle\},$
$\{\langle age, insurance\rangle, \langle matriculation, age\rangle\},$
$\{\langle age, insurance\rangle, \langle matriculation, salary\rangle\},$
$\{\langle age, salary\rangle, \langle matriculation, age\rangle\},$
$\{\langle age, salary\rangle, \langle matriculation, insurance\rangle\}$
$\}.$

□

The notion of *type structural similarity* (*tss*) is formally introduced below.

Definition 5.4. [Type structural similarity (tss)] Consider a weighted expanded form $\mathcal{S}^p = (\Sigma', \mathcal{H}^p_\Sigma)$ of an XML-Schema Σ, and two complexTypes τ_i,τ_j of the schema. Assume $\mathcal{O} = \{elemOf_{min}, elemOf_{Max}, attrOf\}$, $\mathcal{R} \in \mathcal{O}$, and let $P_\mathcal{R} \in \mathcal{C}_\mathcal{R}(\tau_i, \tau_j)$ be one of the candidate sets (bags) of pairs such that the sum of the *ics* of the pairs is maximum, i.e.:

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

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

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

where $m_{j,\mathcal{R}}$ and *typeOf* are defined as in the previous definitions, *ics* is the information content similarity, and $w_\mathcal{R}$ is a weight such that:

$$\sum_{\mathcal{R} \in \mathcal{O}} w_\mathcal{R} \leq 1$$

that is defined by the user (depending on the application domain), in order to give maximum flexibility to the method. Note that $0 \leq tss(\tau_i, \tau_j) \leq 1$. □

Intuitively, in the above definition the maximum among the sums of the *ics* is chosen since it allows us to determine the maximum number of attributes (or elements) that are "common to", or "much similar in" the complexTypes. This will be better clarified by the example below.

Example 5.4. Suppose that $w_{elemOf_{min}} = w_{elemOf_{Max}} = w_{attrOf} = \frac{1}{3}$ and consider again *studentType* and *jobHolderType*. Within the $\mathcal{C}_{attrOf}(studentType, jobHolderType)$ set defined in the previous example, there are two sets such that the sum of the *ics* of the pairs is maximum, that are:

$\{\langle age, age\rangle, \langle matriculation, insurance\rangle\}$,

$\{\langle age, age\rangle, \langle matriculation, salary\rangle\}$,

since, $ics(age, age) = 1$, and $ics(matriculation, insurance) = ics(matriculation, salary) = 0$ (whereas in the other cases the *ics* of both the pairs are null). Of course, in the case of $\mathcal{C}_{elemOf_{min}}(studentType, jobHolderType)$, the selected set of pairs is:

$\{\langle firstName, firstName\rangle, \langle lastName, lastName\rangle\}$,

and in the case of $\mathcal{C}_{elemOf_{Max}}(studentType, jobHolderType)$, the selected bag of pairs is:

$\{\langle firstName, firstName\rangle, \langle firstName, firstName\rangle, \langle lastName, lastName\rangle\}$.

Therefore:

$tss(studentType, jobHolderType) = \frac{1}{3}(\frac{1+1}{3} + \frac{1+1+1}{5} + \frac{1}{3}) = 0.53$.

Let us consider now, for instance, *boyFriendType* and *jobHolderType* (see their expansions in Example 4.3). Their *tss* is defined as follows:

$tss(boyFriendType, jobHolderType) = \frac{1}{3}(\frac{1+1}{3} + \frac{1+1+1}{5} + \frac{1+1+1}{4}) = 0.67$.

In fact, in this case the sets (bags) of pairs with maximum sums within the related $\mathcal{C}_{elemOf_{min}}$, $\mathcal{C}_{elemOf_{Max}}$, and $\mathcal{C}_{attrOf}$ are respectively:

$\{\langle firstName, firstName\rangle, \langle lastName, lastName\rangle\}$

$\{\langle firstName, firstName\rangle, \langle firstName, firstName\rangle, \langle lastName, lastName\rangle\}$

$\{\langle age, age\rangle, \langle indemnity, insurance\rangle, \langle earnings, salary\rangle\}$

since $ics(indemnity, insurance) = ics(earnings, salary) = 1$ because of synonyms, according to the *SynSet* of our running example (see Example 4.4).

□

Note that the *tss* is a partial similarity measure determining only the similarity of typed elements and attributes of complexTypes and, therefore, sometimes it may not comply with the intuition. For this reason, in the next section a combined similarity measure for elements is introduced.

5.3. Combined Similarity

The introduction of the *ics* and *tss* allows us to present the notion of *combined similarity* (*CombSim*) for XML-Schema elements. It is essentially given by the weighted average of the *ics* and the *tss* of their associated types.

Definition 5.5. [Combined Similarity (CombSim)] Consider a weighted expanded form $\mathcal{S}^p = (\Sigma', \mathcal{H}^p_\Sigma)$ of an XML-Schema Σ, a $SynSet_\Sigma$, and two element declarations of the schema:

$e_i : \tau_i$

$e_j : \tau_j$.

The *combined similarity* of e_i,e_j, denoted by $CombSim(e_i, e_j)$, is defined as follows:

$$CombSim(e_i, e_j) = ics(\tau_i, \tau_j) * w + tss(\tau_i, \tau_j) * (1 - w)$$

where *ics* is the information content similarity, *tss* is the type structural similarity, w is a weight such that $0 \leq w \leq 1$, that is defined by the user. Note that $0 \leq CombSim(e_i, e_j) \leq 1$. □

Example 5.5. With the same assumptions of the previous example ($w_{\mathcal{R}} = \frac{1}{3}$), assume $w = \frac{1}{2}$. Let us start by comparing two elements whose types are not related in the hierarchy, as for instance, *student* and $jobHolder$. The following holds:

$$CombSim(student, jobHolder) = ics(studentType, jobHolderType)\tfrac{1}{2} +$$
$$tss(studentType, jobHolderType)\tfrac{1}{2} = \tfrac{1}{2}(0.27 + 0.53) = 0.40$$

since we have seen in Examples 5.1, and 5.4, respectively:

$$ics(studentType, jobHolderType) = 0.27$$

and:

$tss(studentType, jobHolderType) = 0.53.$

The element similarity significantly increases in the case of elements whose types are directly related in the hierarchy, as for instance $jobHolder$, and $employee$. In fact, the following holds:

$CombSim(jobHolder, employee) = \frac{1}{2}(0.78 + 0.78) = 0.78.$

In the case of elements whose types are in hierarchy, but they are not directly related, as for instance $jobHolder$ and $worker$, we have:

$CombSim(jobHolder, worker) = \frac{1}{2}(0.67 + 0.76) = 0.72$

since:

$ics(jobHolderType, workerType) = \frac{2 \log p(workerType)}{\log p(jobHolderType) + \log p(workerType)} = 0.67,$

and:

$tss(jobHolderType, workerType) = \frac{1}{3}(\frac{1+1+0.93}{3} + \frac{1+1+1+0.93+0.93}{5} + \frac{1}{3}) = 0.76.$

Note that the tss above has been computed taking into account the ics of the types associated with the element $occupation$ in $jobHolderType$ and $workerType$ that are, respectively, $jobType$ and $activityType$. In particular, the following holds:

$ics(jobType, activityType) = \frac{2 \log p(activityType)}{\log p(activityType) + \log p(jobType)} = 0.93.$

Just to give some more examples, let us compare $collegian$ with $boyFriend$ and, successively, with $girlFriend$. The following holds:

$CombSim(collegian, boyFriend) = \frac{1}{2}(0.22 + 0.75) = 0.49$

whereas, in the case of $collegian$ and $girlFriend$ the tss, and therefore the element similarity, increases due to the presence of the pair of synonyms $college$ and $educationInstitute$:

$CombSim(collegian, girlFriend) = \frac{1}{2}(0.22 + 0.89) = 0.56.$

Before concluding, it is interesting to analyze job and $boyFriend$, whose types have $anyType$ as lub in the type hierarchy, and whose type declarations share two synonyms (namely, pay and $earnings$, respectively). In this case, the element similarity significantly decreases, as shown below:

$CombSim(job, boyFriend) = \frac{1}{2}(0 + 0.08) = 0.04.$

These similarity scores will be discussed in the next section.

□

6. Illustrative Data Set

In this section the similarity among some of the elements of our running example is determined according to Resnik [13], Wu&Palmer [21], Lin [5] and, furthermore, *SymOntos* [16], and *Dice* [18]. The results are then compared with the element similarity (*CombSim*) measure proposed in this paper (see Table 2). Since the proposals of Resnik, Lin, and *SymOntos* have already been discussed, below the approaches of Wu&Palmer, and *Dice* are briefly recalled.

6.1. Wu&Palmer and Dice Approaches

According to Wu&Palmer [21], given two concepts n_1, n_2 in a taxonomy, their similarity ($Sim_{WP}(n_1, n_2)$) is defined as:

$Sim_{WP}(n_1, n_2) = \frac{2d_3}{d_1+d_2+2d_3}$

where d_1 and d_2 are the lengths of the paths in the ISA taxonomy from n_1 and n_2, respectively, to their most specific common superconcept, say n_3 (that corresponds to the concept providing the maximum information content shared by n_1, and n_2), and d_3 is the length of the path from n_3 to the root of the taxonomy. For instance, suppose we want to compare the *student* and $jobHolder$ elements, and assume that $n_1 = studentType$ and $n_2 = jobHolderType$. In this case, according to Figure 1, their most specific common superconcept is provided by $personType$. Therefore, $d_1 = 1$, $d_2 = 3$, and $d_3 = 1$ and $Sim_{WP}(student, jobHolder) = 0.33$.

According to *Dice*'s function [18], given two concepts, say n_1, and n_2, each described by a set of features (for instance, a set of attributes or a sequence of elements), say $F(n_1)$, $F(n_2)$, respectively, their similarity ($Sim_{Dice}(n_1, n_2)$) is defined as follows:

Table 2: Comparison among different similarity measures

	Resnik	*Wu&Palmer*	*Lin*	*SymOntos*	*Dice*	*tss*	*CombSim*
(*student*, *jobHolder*)	3.61	0.33	0.27	0.53	0.65	0.53	0.40
(*employee*, *jobHolder*)	10.60	0.86	0.78	0.78	0.83	0.78	0.78
(*worker*, *jobHolder*)	8.26	0.67	0.67	0.78	0.83	0.76	0.72
(*collegian*, *boyFriend*)	3.61	0.33	0.22	0.75	0.76	0.75	0.49
(*collegian*, *girlFriend*)	3.61	0.33	0.22	0.89	0.89	0.89	0.56
(*lover*, *girlFriend*)	13.62	0.80	0.91	0.98	0.93	0.89	0.90
(*job*,*boyFriend*)	0	0	0	0.08	0.10	0.08	0.04

$$Sim_{Dice}(n_1, n_2) = \frac{2|A(n_1,n_2)|}{|F(n_1)|+|F(n_2)|}$$

where:

$$A(n_1, n_2) = \{(a, b) \mid a \in F(n_1), b \in F(n_2), (a, b) \in Aff(n_1, n_2)\}$$

$Aff(n_1, n_2)$ is the set of pairs of features of n_1 and n_2 showing affinity according to some criteria similar to the maximum weighted matching problem in bipartite graphs, and $|A(n_1, n_2)|$, $|F(n_1)|$, and $|F(n_2)|$ are the cardinalities of the sets $A(n_1, n_2)$, $F(n_1)$, and $F(n_2)$, respectively. In addition, since we deal with XML-Schemas, elements with minimum numbers of occurrences, elements with maximum numbers of occurrences, and attributes are compared separately and, successively, the average of the obtained values is performed (in line with the assumptions made for the other approaches). For instance, in the case of *student* and *jobHolder*, we have:

$$Sim_{Dice}(student, jobHolder) = \frac{1}{3}(\frac{2*2}{2+3} + \frac{2*3}{3+5} + \frac{2*1}{2+3}) = 0.65$$

6.2. Evaluation

The data set defined in Table 2 has been given to show that, in the literature, there exist similarity methods that mainly focus on the taxonomic organization of the concepts, while giving less (or no) importance to the features (structures) of the concepts or, viceversa, there are methods addressing the similarity of the sets of the features of the concepts, while giving a marginal importance to the concept taxonomy. In the case of the representative proposals that have been selected here, Resnik, Wu&Palmer and Lin address the hierarchy, whereas *SymOntos* and *Dice* mainly focus on the structures. It is worth noting that the *CombSim* proposal relies on the experimental results defined in the literature, and Table 2 does not intend to validate our method but to show the gap among the different approaches and how this gap can be reduced according to this proposal.

Before illustrating the similarity scores defined in Table 2, some considerations have to be done starting with the problem about "ideal" values. In general, ideal values are established according to human-subject experiments and, in the literature, the similarity scores assigned by human subjects in the Miller&Charles experiments are often addressed [43]. In these experiments, 28 pairs of nouns have been given to 38 undergraduate subjects, and one score - on a scale of 0 to 4 - has been assigned to each pair. By using the Miller&Charles experiments, it has already been shown in [5], and also by other authors, see for instance in [14], that Lin's approach shows a higher correlation with human judgement than the other two hierarchy-based methods, i.e., Resnik[2] and Wu&Palmer. However, in different contexts the structural approach

[2]Note that in Table 2 Resnik's similarity scores have been given just to show that in this case concept similarity depends only on the shared maximum information content, and it is independent of the comparing concepts. See, for instance, the pairs ($student$,$jobHolder$), ($collegian$,$boyFriend$), and ($collegian$,$girlFriend$), whose similarity scores are all equal to 3.61 since they all share the same *lub* represented by $person$.

is preferred which takes into account the heterogeneity of concept features. Examples include conceptual schema analysis, information sharing from multiple heterogeneous sources, intelligent information integration, and Web service discovery [6, 17]. In particular, the *Dice*'s function is adopted, whose similarity scores, in most of cases, are significantly different from the ones obtained according to Lin (see for instance Table 2). The *CombSim* measure proposed in this paper aims at reducing the distances between these two approaches by making a weighted average between Lin and the structural approach. Note that, the structural approach (*tss*) defined in *CombSim* is an extension of that of *SymOntos*, which has also been preferred to *Dice* since *tss* allows concept similarity to be computed by explicitly addressing the similarity degrees of the concept components and the related types. For instance, consider in Table 2 the similarity scores obtained for the pairs (*employee*,*jobHolder*) and (*worker*,*jobHolder*). *EmployeeType* and *workerType* differ only for the types associated with the *occupation* element, that are *jobType* and *activityType*, respectively. Both *SymOntos* and *Dice* do not allow to capture this distinction and, in fact, the similarity scores of (*employee*,*jobHolder*) and (*worker*,*jobHolder*) coincide (they are equal to 0.78 in the case of *SymOntos*, and 0.83 in the case of *Dice*). This is not the case of *CombSim* where, as expected according to the types associated with *occupation*, the *tss* of the pair (*employee*,*jobHolder*) is greater than that of (*worker*,*jobHolder*) (i.e., 0.78 and 0.76, respectively). Then, the combination with *Lin* leads to the different similarity scores 0.78 and 0.72, respectively.

Let us now analyze some of the pairs of elements of Table 2 in details. Consider (*collegian*, *boyFriend*) and (*collegian*, *girlFriend*). By addressing the hierarchical organization of concepts, their similarity scores coincide according to all three hierarchy-based methods, i.e., Resnik, Wu&Palmer, Lin. In particular, they also coincide according to Lin due to the same shared information content (*person*) and, in addition, the same frequencies of *boyFriend* and *girlFriend*.

By addressing their structures as performed in *SymOntos*, *Dice*, or *tss*, their scores differ having *boyFriend* and *girlFriend* different sets of attributes, namely *indemnity* and *earnings* vs *educationInstitute*, respectively. In particular, the similarity scores of the pair (*collegian*, *girlFriend*) are greater than those of the pair (*collegian*, *boyFriend*) due to an additional pair of matching attributes. In fact, in the latter case, the types have only one attribute in common represented by *age*, whereas in the former case, in addition to *age*, also the attributes *college* (of *collegianType*) and *educationInstitute* (of *girlFriendType*), which are synonyms, are present. Therefore, by the definition, also in the case of *CombSim* the similarity scores of these two pairs of elements differ.

Consider now (*student*, *jobHolder*) and (*employee*, *jobHolder*). According to Lin, *student* and *jobHolder* share the information content provided by the very general concept *person*, and their similarity score is quite low (0.27). In the case of *employee* and *jobHolder*, the similarity significantly increases (0.78) since they share the information content provided by *employeeType* which is the base type of *jobHolderType* (and the information content of *employeeType* is significantly greater than that of *personType*). According to both *SymOntos* and *Dice*, the pair (*student*, *jobHolder*) shows higher similarity scores than that of Lin due to their features. In fact, they share a single occurrence of both *firstName* and *lastName* (in the case of $elemOf_{min}$), a double occurrence of *firstName* and a single occurrence of *lastName* (in the case of $elemOf_{Max}$), and the attribute *age* (in the case of *attrOf*). If we consider (*employee*, *jobHolder*), the similarity scores are even greater than the ones of (*student*, *jobHolder*) since *employeeType* is the base type of *jobHolderType* (therefore we have an inclusion of the related elements and attributes, respectively).

Let us now analyze the pair (*job*, *boyFriend*). On one hand, they share no information content according to the hierarchy, therefore the Lin's score is null. On the other hand, *jobType* and

Table 3: Evaluation summary

	Resnik	*Wu&Palmer*	*Lin*	*SymOntos*	*Dice*	*tss*	*CombSim*
hierarchy-based approach	X	X	X				X
feature-based appr.				X	X	X	X
sim. degree of comp. (comp.)				X		X	X
sim. degree of nested types						X	X
sim. increases with the n. of sim. comp.				X	X	X	X
shared information content (ic) only	X						
shared ic and ic of comparing concepts			X				X

boyFriendType have *pay* and *earnings* attributes, respectively, which are synonyms (see the *SynSet* in Example 4.4). Therefore, according to both *SymOntos* and *Dice* we have non-null, although low, values (0.08 and 0.10, respectively). According to *CombSim* an approximately average value (0.04) is obtained between Lin and *SymOntos* or Lin and *Dice*. Note that one might expect that *job* and *boyFriend* are not similar at all, i.e., their similarity score is null, although they have common features. For this reason according to *CombSim*, the user has the possibility of assigning more or less importance to the sets of features (or the hierarchical component) by properly specifying the related weights.

In Table 3 a summary about the comparison among the methods addressed in this paper is given. It is worth recalling that within the approaches proposed by Resnik, Wu&Palmer, and Lin, that have been conceived to determine semantic similarity of hierarchically organized concepts (without addressing the structures), the experimental results defined in the literature show a higher correlation of Lin's approach with human judgement. However, in various contexts, often the structural (or featural) approach is adopted, whose similarity scores significantly differ

from Lin. In order to reduce the distances between these approaches, the element similarity (*CombSim*) has been proposed which addresses both the taxonomy (according to the *ics*) and the concept structures (according to the *tss*). We remark that, as shown in Table 3, the *tss* (and therefore *CombSim*) overcomes the limitations of both *SymOntos* and *Dice* by addressing the similarity degrees of concept typed components.

7. Conclusion and Future Work

In this paper a method for comparing XML-Schema elements has been presented. This proposal combines and revisits the probabilistic approach referred to as the *information content* approach, and a method for comparing feature vectors inspired by the *maximum weighted matching* problem in bipartite graphs. As a future work, we aim at extending the method to wider XML-Schemas including *choice*, *all*, and derived SimpleTypes. Furthermore, it could be interesting to integrate this proposal with that presented in [24], based on domain ontologies.

References

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

[2] Berners-Lee, T. et al. (2001) *The Semantic Web*; Scientific American.

[3] Ding, Y. and Fensel, D. and Klein, M. and Omelayenko, B. (2002) *The semantic web: yet another hip?*; Data & Knowledge Engineering, 41(2-3), 205-227.

[4] Rodriguez, A. and Egenhofer, M. (2004) *Comparing Geospatial Entity Classes: An Asymmetric and Context-Dependent Similarity Measure*; Int. Journal of Geographical Information Science (IJGIS), 18(3), 229–256.

[5] Lin, D. (1998) *An Information-Theoretic Definition of Similarity*; Proc. of the Int. Conference on Machine Learning (ICML), Madison, Wisconson, USA, July 24-27, pp.296-304, Morgan Kaufmann.

[6] Bianchini, D. and De Antonellis, V. and Melchiori, M. (2005) *Capability Matching and Similarity Reasoning in Service Discovery*; Missikoff M. and De Nicola A. (Eds.), Proc. of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability (EMOI-INTEROP), 13-14 June, pp.285-296, Porto, Portugal, CEUR-WS.org.

[7] Su, H. and Kuno, H. and Rundensteiner, E.A. (2001) *Automating the Transformation of XML Documents*; Proc. of 3rd Int. Workshop on Web Information and Data Management (WIDM), 9 November, pp.68-75, Atlanta, Georgia, ACM.

[8] Yi, S. and Huang, B. and Chan, W.T. (2005) *XML application schema matching using similarity measure and relaxation labeling*; Information Sciences 169(1-2), 27-46.

[9] Stroulia, E. and Wang, Y. (2005) *Structural and Semantic Matching for Assessing Web-service Similarity*; Int. Journal of Cooperative Information Systems, 14(4), 407-438.

[10] Lee, J.H. and Kim, M.H. and Lee, Y.J. (1993) *Information Retrieval Based on Conceptual Distance in IS-A Hierarchies*; Journal of Documentation, 49(2), 188-207.

[11] Rada, L. and Mili, H. and Bicknell, E. and Bletter, M. (1989) *Development and application of a metric on semantic nets*; IEEE Transaction on systems, Man, and Cybernetics, 19(1), 17-30.

[12] Resnik, P. (1995) *Using Information Content to Evaluate Semantic Similarity in a Taxonomy*; Proc. of the Int. Joint Conference on Artificial Intelligence (IJCAI), Montréal, Canada, August 20-25, pp.448-453, Morgan Kaufmann.

[13] Resnik, P. (1999) *Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language*; J. of Artificial Intelligence Reserach (JAIR), 11, 95-130.

[14] Jiang, J. J. and Conrath, D. W. (1997) *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy*; The Computing Research Repository (CoRR), cmp-lg/9709008.

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

[16] Formica, A. and Missikoff, M. (2002) *Concept Similarity in SymOntos: an Enterprise Ontology Management Tool*; The Computer Journal, 45(6), 583-594.

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

[18] Maarek, Y.S. and Berry, D.M. and Kaiser, G.E. (1991) *An Information Retrieval Approach For Automatically Constructing Software Libraries*; IEEE Transactions on Software Engineering, 17(8), 800-813.

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

[20] 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.

[21] Wu, Z. and Palmer, M. (1994) *Verb semantics and lexical selection*; Proc. of the 32nd Annual Meeting of the Associations for Computational Linguistics; June 27-30, pp.133-138, Las Cruces, New Mexico.

[22] Spanoudakis, G. and Constantopoulos, P. (1994) *Similarity for Analogical Software Reuse: A Computational Model*; Proc. of the Eleventh European Conference on Artificial Intelligence, Amsterdam, The Netherlands, August 8-12, pp.18-22, John Wiley&Sons, New York.

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

[24] Formica, A. (2005) *Similarity of XML-Schema Elements supported by Domain Ontologies*; International Journal of Software Engineering and Knowledge Engineering (IJSEKE), 15(1), 117-130.

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

[26] *WordNet 2.1: A lexical database for the english language* (2005) http://www.cogsci.princeton.edu/cgi-bin/webwn.

[27] Ziegler, P. and Kiefer, C. and Sturm, C. and Dittrich, K.R. and Bernstein, A. (2006) *Detecting Similarities in Ontologies with the SOQA-SimPack Toolkit*; Proc. of International Conference on Extending Database Technology (EDBT), 26-31 March 2006, pp.59-76, Munich, Germany.

[28] Bex, G.J. and Neven, F. and Bussche, J. (2004) *DTDs versus XML Schema: A Practical Study*; Proc. of International Workshop on the Web and Databases (WebDB), S.Amer-Yahia and L.Gravano (Eds.), Maison de la Chimie, Paris, France, June 17-18, pp.79-84.

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

[30] Mani, M. and Lee, D. and Muntz, R.R. (2001) *Semantic Data Modeling Using XML Schemas*; Int. Conference on Conceptual Modeling (ER), H.S.Kunii, S.Jajodia and A.Sølvberg (Eds.), November 27-30, pp.149-163, Yokohama, Japan, Lecture Notes in Computer Science (LNCS) 2224, Springer.

[31] Formica, A. (2004) *Legality of XML-Schema Type Hierarchies*; The Computer Journal, 47(5), 591-601.

[32] Wang, Y. and DeWitt, D.J. and Cai J. (2003) *X-Diff: An Effective Change Detection Algorithm for XML Documents*; Proc. of the 19th Int. Conference on Data Engineering (ICDE); U.Dayal, K.Ramamritham, and T.M.Vijayaraman (Eds.), March 5-8, pp.519-530, Bangalore, India, IEEE Computer Society.

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

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

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

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

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

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

[39] Francis, W. N. and Kucera, H. (1982) *Frequency Analysis of English Usage*, Houghton Mifflin, Boston.

[40] Fellbaum C. (1998) *A Semantic Network of English: the Mother of all WordNets*; Computers and the Humanities 32, 209-220.

[41] Ross S. (1976) *A First Course in Probability*; Macmillan.

[42] Formica, A. (2003) *Satisfiability of Object-Oriented Database Constraints with Bag and Set Attributes*; Information Systems, 28(3), 213-224.

[43] Miller, G.A. and Charles, W.G. (1991) *Contextual correlates of semantic similarity*; Language and Cognitive Processes, 6(1), 1-28.