

“Almost automatic” and semantic integration of XML Schemas at various “severity” levels

Pasquale De Meo¹, Giovanni Quattrone¹,
Giorgio Terracina², and Domenico Ursino¹

¹ DIMET, Università Mediterranea di Reggio Calabria, Via Graziella, Località Feo di Vito, 89060 Reggio Calabria, Italy,

² Dipartimento di Matematica, Università della Calabria, Via Pietro Bucci, 87036 Rende (CS), Italy

demeo@ing.unirc.it, quattrone@ing.unirc.it,
terracina@mat.unical.it, ursino@unirc.it

Abstract. This paper presents a novel approach for the integration of a set of XML Schemas. The proposed approach is specialized for XML, is almost automatic, semantic and “light”. As a further, original, peculiarity, it is parametric w.r.t. a “severity” level against which the integration task is performed. The paper describes the approach in all details, illustrates various theoretical results, presents the experiments we have performed for testing it and, finally, compares it with various related approaches already proposed in the literature.

1 Introduction

The Web is presently playing a key role for both the publication and the exchange of information among organizations. As a matter of fact, it is becoming the reference infrastructure for most of the applications conceived to handle interoperability among partners.

In order to make Web activities easier, W3C (World Wide Web Consortium) proposed XML (eXtensible Markup Language) as a new standard information exchange language that unifies representation capabilities, typical of HTML, and data management features, typical of classical DBMS.

The twofold nature of XML allowed it to gain a great success and, presently, most of the new documents published on the Web are written in XML. However, from the data management point of view, XML documents alone have limited and primitive capabilities. In order to improve these capabilities, in such a way to make them similar to those typical of classical DBMS, W3C proposed to associate XML Schemas with XML documents. An XML Schema can be considered as a sort of catalogue of the information typologies that can be found in the corresponding XML documents; from another point of view, an XML Schema defines a reference context for the corresponding XML documents.

Certainly, XML exploitation is a key step for improving the interoperability of Web information sources; however, that alone is not enough to completely fulfill such a task. Indeed, the heterogeneity of data exchanged over the Web regards

not only their formats but also their semantics. The use of XML allows format heterogeneity to be faced; the exploitation of XML Schemas allows the definition of a reference context for exchanged data and is a first step for handling semantic diversities; however, for a complete and satisfactory management of these last, an integration activity is necessary.

This paper provides a contribution in this setting and proposes an approach for the integration of a set of XML Schemas. Our approach behaves as follows: first it determines interscheme properties [2,7,11,17,15], i.e., terminological and structural relationships holding among attributes and elements belonging to involved XML Schemas. After this, some of the derived properties are exploited for modifying involved Schemas in order to make them structurally and semantically uniform. The modified Schemas are, finally, integrated for obtaining the global Schema.

Let us now examine the peculiarities of our approach in more detail. First, it has been specifically conceived for operating on XML sources. In this sense it differs from many other approaches already presented in the literature which integrate information sources having different formats and structure degrees (e.g., relational databases, XML documents, object-oriented sources and so on). Generally, such approaches translate all involved information sources into a common representation and, then, carry out the integration activity. On the contrary, our approach is specialized for integrating XML Schemas. With regard to this, it is worth pointing out that: *(i)* the integration of XML Schemas will play a more and more relevant role in the future; *(ii)* the exploitation of generic approaches designed to operate on information sources with different formats, for performing the integration of a set of XML Schemas (i.e., a set of sources having the same format), is unnecessarily expensive and inefficient. Indeed, it would require the translation of involved XML Schemas in another format and the translation of the integrated source from such a format back to XML.

Our approach is almost automatic; in this sense it follows the present trend relative to integration techniques. Indeed, owing to the enormous increase of the number of available information sources, all integration approaches proposed in the last years are semi-automatic; generally, they require the human intervention for both a pre-processing phase and the validation of obtained results. The overwhelming amount of sources available on the Web leads each integration task to operate on a great number of sources; this requires a further effort for conceiving more automatic approaches. The approach we are proposing here provides a contribution in this setting since it is almost automatic and requires the user intervention only for validating obtained results.

Our approach is “light”; with regard to this we observe that most of the existing approaches are quite complex, based on a variety of thresholds, weights, parameters and so on; they are very precise but difficult to be applied and fine tuned when involved sources are numerous, complex and belonging to heterogeneous contexts. Our approach does not exploit any threshold or weight; as a consequence, it is simple and light, since it does not need a tuning activity.

Our approach is semantic in that it follows the general trend to take into account the semantics of concepts belonging to involved information sources during the integration task [2,4,7,11]. Given two concepts belonging to different information sources, one of the most common way for determining their semantics consists of examining their neighborhoods since the concepts and the relationships which they are involved in contribute to define their meaning. As a consequence, two concepts, belonging to different information sources, are considered semantically similar and are merged in the integrated source if their neighborhoods are similar.

We argue that all the peculiarities we have examined above are extremely important for a novel approach devoted to integrate XML Schemas. However, the approach we are proposing here is characterized by a further feature that, in our opinion, is extremely innovative and promising; more specifically, it allows the choice of the “severity level” against which the integration task is performed. Such a feature derives from the consideration that applications and scenarios possibly benefiting of an integration task on the Web are numerous and extremely various. In some situations (e.g., in Public Administrations, Finance and so on) the integration process must be very severe in that two concepts must be merged only if they are strongly similar; in such a case a high severity degree is required. In other situations (e.g., tourist Web pages) the integration task can be looser and can decide to merge two concepts having some similarities but presenting also some differences. At the beginning of the integration activity our approach asks the user to specify the desired “severity” degree; this is the only information required to her/him until the end of the integration task, when she/he has to validate obtained results. It is worth pointing out that, to the best of our knowledge, no approaches handling the information source integration at various “severity” levels have been previously presented in the literature. Interestingly enough, a classical approach can be seen as a particular case of that presented in this paper in which a severity level is fixed and all concept merges are performed w.r.t. this level.

2 Neighborhood Construction

In this section we formally introduce the concept of neighborhood of an element or an attribute of an XML Schema. As pointed out in the Introduction, this concept plays a key role in the various algorithms which our approach consists of. Preliminarily we introduce the concept of *x-component* which allows both elements and attributes of an XML document to be uniformly handled.

Definition 1. Let S be an XML Schema; an *x-component* of S is either an element or an attribute of S . \square

An *x-component* is characterized by its name, its typology (indicating if it is either a complex element or a simple element or an attribute) and its data type.

Definition 2. Let S be an XML Schema; the set of its *x-components* is denoted as $XCompSet(S)$. \square

We introduce now some boolean functions that allow to determine the strength of the relationship existing between two x-components x_S and x_T of an XML Schema S . They will be exploited for deriving interscheme properties and, ultimately, for integrating XML Schemas. The functions are:

- $veryclose(x_S, x_T)$, that returns *true* if and only if: (i) $x_T = x_S$, or (ii) x_T is an attribute of x_S , or (iii) x_T is a simple sub-element of x_S ;
- $close(x_S, x_T)$, that returns *true* if and only if (i) x_T is a complex sub-element of x_S , or (ii) x_T is an element of S and x_S has an *IDREF* or an *IDREFS* attribute referring x_T ;
- $near(x_S, x_T)$, that returns *true* if and only if either $veryclose(x_S, x_T) = true$ or $close(x_S, x_T) = true$; in all the other cases it returns *false*;
- $reachable(x_S, x_T)$, that returns *true* if and only if there exists a sequence of *distinct* x-components x_1, x_2, \dots, x_n such that $x_S = x_1, near(x_1, x_2) = near(x_2, x_3) = \dots = near(x_{n-1}, x_n) = true, x_n = x_T$. \square

We are now able to compute the connection cost from x_S to x_T .

Definition 3. Let S be an XML Schema and let x_S and x_T be two x-components of S . The Connection Cost from x_S to x_T , denoted by $CC(x_S, x_T)$, is defined as:

$$CC(x_S, x_T) = \begin{cases} 0 & \text{if } veryclose(x_S, x_T) = true \\ 1 & \text{if } close(x_S, x_T) = true \\ \mathcal{C}_{ST} & \text{if } reachable(x_S, x_T) = true \text{ and } near(x_S, x_T) = false \\ \infty & \text{if } reachable(x_S, x_T) = false \end{cases}$$

where $\mathcal{C}_{ST} = \min_{x_A} (CC(x_S, x_A) + CC(x_A, x_T))$ for each x_A such that $reachable(x_S, x_A) = reachable(x_A, x_T) = true$.

We are now provided with all tools necessary to define the concept of neighborhood of an x-component.

Definition 4. Let S be an XML Schema and let x_S be an x-component of S . The j^{th} neighborhood of x_S is defined as: \square

$$neighborhood(x_S, j) = \{x_T \mid x_T \in XCompSet(S), CC(x_S, x_T) \leq j\}$$

The construction of all neighborhoods can be easily carried out with the support of the data structure introduced in the next definition.

Definition 5. Let D be an XML document and let S be the corresponding XML Schema. The *XS-Graph* relative to S and D is an oriented labeled graph defined as $XG(S, D) = \langle N(S), A(S, D) \rangle$. Here, $N(S)$ is the set of nodes of $XG(S, D)$; there is a node in $XG(S, D)$ for each x-component of S . $A(S, D)$ is the set of arcs of $XG(S, D)$; there is an arc $\langle N_S, N_T, f_{ST} \rangle$ in $XG(S, D)$ for each pair (x_S, x_T) such that $near(x_S, x_T) = true$; in particular, N_S (resp., N_T) is the node of $XG(S, D)$ corresponding to x_S (resp., x_T) and $f_{ST} = CC(x_S, x_T)$. \square

The following proposition measures the computational complexity of the construction of $XG(S, D)$.

Proposition 1. Let D be an XML document and let S be the corresponding XML Schema. Let n be the number of x-components of S and let N_{inst} be the number of instances of D . The worst case time complexity for constructing $XG(S, D)$ from S and D is $O(\max\{n, N_{inst}^2\})$. \square

With regard to this result we observe that, in an XML document, in order to determine the element which an IDREFS attribute refers to, it is necessary to examine the document, since neither the DTD nor the XML Schema provide such an information. As a consequence, the dependency of the computational complexity from N_{inst} cannot be avoided. However, we point out that the quadratic dependency from N_{inst} is mainly a theoretical result; indeed, it derives from the consideration that each IDREFS attribute could refer to N_{inst} components. Actually, in real situations, each IDREFS attribute refers to a very limited number of instances; as a consequence, the dependency of the computational complexity from N_{inst} is generally linear.

The next theorem determines the worst case time complexity for computing all neighborhoods of all x-components of an XML Schema S .

Theorem 1. Let $XG(S, D)$ be the XS-Graph associated with an XML document D and an XML Schema S and let n be the number of x-components of S . The worst case time complexity for computing all neighborhoods of all x-components of S is $O(n^3)$. \square

Example 1. Consider the XML Schema S_1 , shown in Figure 1, representing a shop. Here *customer* is an x-component and its typology is “complex element” since it is an element declared with a “complex type”. Analogously *SSN* is an x-component, its typology is “attribute” and its data type is “string”. All the other x-components of S_1 , the corresponding typologies and data types can be determined similarly.

In S_1 , $veryclose(customer, firstName) = true$ because *firstName* is a simple sub-element of *customer*; analogously $veryclose(customer, SSN) = true$ and $close(customer, musicAcquirement) = true$. As for neighborhoods, we have that:

$$neighborhood(customer, 0) = \{customer, SSN, firstName, lastName, address, gender, birthDate, profession\}$$

All the other neighborhoods can be determined similarly. \square

3 Extraction of interscheme properties

In this section we illustrate an approach for computing interscheme properties among x-components belonging to different XML Schemas. As pointed out in the Introduction, their knowledge is crucial for the integration task. The interscheme properties considered in this paper are *synonymies* and *homonymies*. Given two x-components x_A and x_B belonging to different XML Schemas, a *synonymy* between x_A and x_B indicates that they represent the same concept; an *homonymy*

```

<?xml version="1.0" encoding="UTF-8"?> <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
<!-- Definition of attributes -->
<xs:attribute name="SSN" type="xs:string"/>
<xs:attribute name="code" type="xs:ID"/>
<xs:attribute name="acquiredBooks" type="xs:IDREFS"/>
<xs:attribute name="acquiredMusica" type="xs:IDREFS"/>
<xs:attribute name="acquirementDate" type="xs:date"/>
<!-- Definition of simple elements -->
<xs:element name="firstName" type="xs:string"/>
<xs:element name="lastName" type="xs:string"/>
<xs:element name="address" type="xs:string"/>
<xs:element name="gender" type="xs:string"/>
<xs:element name="birthDate" type="xs:date"/>
<xs:element name="profession" type="xs:string"/>
<xs:element name="artist" type="xs:string"/>
<xs:element name="author" type="xs:string"/>
<xs:element name="title" type="xs:string"/>
<xs:element name="pubYear" type="xs:integer"/>
<xs:element name="publisher" type="xs:string"/>
<xs:element name="genre" type="xs:string"/>
<xs:element name="support" type="xs:string"/>
<!-- Definition of complex elements -->
<xs:element name="bookAcquirement">
  <xs:complexType>
    <xs:attribute ref="acquirementDate"/>
    <xs:attribute ref="acquiredBooks"/>
  </xs:complexType>
</xs:element>
<xs:element name="musicAcquirement">
  <xs:complexType>
    <xs:attribute ref="acquirementDate"/>
    <xs:attribute ref="acquiredMusica"/>
  </xs:complexType>
</xs:element>
<xs:element name="customer">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="firstName"/>
      <xs:element ref="lastName"/>
      <xs:element ref="address"/>
      <xs:element ref="gender"/>
      <xs:element ref="birthDate"/>
      <xs:element ref="profession"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element ref="bookAcquirement"
  minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="musicAcquirement"
  minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute ref="SSN" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="music">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="artist" maxOccurs="unbounded"/>
      <xs:element ref="title"/>
      <xs:element ref="pubYear"/>
      <xs:element ref="genre"/>
      <xs:element ref="support"/>
    </xs:sequence>
    <xs:attribute ref="code" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="book">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="author" maxOccurs="unbounded"/>
      <xs:element ref="title"/>
      <xs:element ref="publisher"/>
      <xs:element ref="pubYear"/>
      <xs:element ref="genre"/>
    </xs:sequence>
    <xs:attribute ref="code" use="required"/>
  </xs:complexType>
</xs:element>
<!-- Definition of root element -->
<xs:element name="shop">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="customer" maxOccurs="unbounded"/>
      <xs:element ref="music" maxOccurs="unbounded"/>
      <xs:element ref="book" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Fig. 1. The XML Schema S_1

between x_A and x_B denotes that they indicate different concepts yet having the same name.

Our technique for computing interscheme properties is semantic [2,7,15] in that, in order to determine the meaning of an x-component, it examines the “context” which it has been defined in. It requires the presence of a thesaurus storing lexical synonymies existing among the terms of a language. In particular, it exploits the English language and WordNet¹ [14]. The technique first extracts all synonymies and, then, exploits them for deriving homonymies.

3.1 Derivation of synonymies

As previously pointed out, in order to verify if two x-components x_{1_j} , belonging to an XML Schema S_1 , and x_{2_k} , belonging to an XML Schema S_2 , are synonymous, it is necessary to examine their neighborhoods. In particular, our approach operates as follows.

First it considers $neighborhood(x_{1_j}, 0)$ and $neighborhood(x_{2_k}, 0)$ and determines if they are similar. This decision is made by computing the objec-

¹ Actually, in the prototype implementing our technique, WordNet is accessed by a suitable API.

tive function associated with the maximum weight matching of a suitable bipartite graph constructed from the x-components of $neighborhood(x_{1_j}, 0)$ and $neighborhood(x_{2_k}, 0)$ and their lexical synonymies as stored in the thesaurus (see below for all details). If $neighborhood(x_{1_j}, 0)$ and $neighborhood(x_{2_k}, 0)$ are similar it is possible to conclude that x_{1_j} and x_{2_k} are *synonymous* [2,15]. However, observe that $neighborhood(x_{1_j}, 0)$ (resp., $neighborhood(x_{2_k}, 0)$) takes into account only attributes and simple elements of x_{1_j} (resp., x_{2_k}); therefore, it considers quite a limited context. As a consequence, the synonymy between x_{1_j} and x_{2_k} derived in this case is more “syntactic” than “semantic” [7,2,15].

If we need a more “severe” level of synonymy detection it is necessary to require not only the similarity of $neighborhood(x_{1_j}, 0)$ and $neighborhood(x_{2_k}, 0)$ but also that of the other neighborhoods of x_{1_j} and x_{2_k} . More specifically, it is possible to introduce a “severity” level u at which synonymies are derived and to say that x_{1_j} and x_{2_k} are synonymous with severity level equal to u if $neighborhood(x_{1_j}, v)$ is similar to $neighborhood(x_{2_k}, v)$ for each v less than or equal to u . The following proposition states an upper bound to the severity level that can be specified for x-component synonymy derivation.

Proposition 2. Let S_1 and S_2 be two XML documents; let x_{1_j} (resp., x_{2_k}) be an x-component of S_1 (resp., S_2); finally, let m be the maximum between the number of complex elements of S_1 and S_2 . The maximum severity level possibly existing for the synonymy between x_{1_j} and x_{2_k} is $m - 1$. \square

A function *synonymous* can be defined which receives two x-components x_{1_j} and x_{2_k} and an integer u and returns *true* if x_{1_j} and x_{2_k} are synonymous with a severity level equal to u , *false* otherwise.

As previously pointed out, computing the synonymy between two x-components x_{1_j} and x_{2_k} implies determining when two neighborhoods are similar. In order to carry out such a task, it is necessary to compute the objective function associated with the maximum weight matching relative to a specific bipartite graph obtained from the x-components of the neighborhoods into consideration.

More specifically, let $BG(x_{1_j}, x_{2_k}, u) = \langle N(x_{1_j}, x_{2_k}, u), A(x_{1_j}, x_{2_k}, u) \rangle$ be the bipartite graph associated with $neighborhood(x_{1_j}, u)$ and $neighborhood(x_{2_k}, u)$ (in the following we shall use the notation $BG(u)$ instead of $BG(x_{1_j}, x_{2_k}, u)$ when this is not confusing). In $BG(u)$, $N(u) = P(u) \cup Q(u)$ represents the set of nodes; there is a node in $P(u)$ (resp., $Q(u)$) for each x-component of $neighborhood(x_{1_j}, u)$ (resp., $neighborhood(x_{2_k}, u)$). $A(u)$ is the set of arcs; there is an arc between $p_e \in P(u)$ and $q_f \in Q(u)$ if a synonymy between the names of the x-components associated with p_e and q_f holds in the reference thesaurus. The maximum weight matching for $BG(u)$ is a set $A'(u) \subseteq A(u)$ of edges such that, for each node $x \in P(u) \cup Q(u)$, there is at most one edge of $A'(u)$ incident onto x and $|A'(u)|$ is maximum (for algorithms solving the maximum weight matching problem, see [8]). The objective function we associate with the maximum weight matching is $\phi_{BG}(u) = \frac{2|A'(u)|}{|P(u)| + |Q(u)|}$.

We assume that if $\phi_{BG}(u) > \frac{1}{2}$ then $neighborhood(x_{1_j}, u)$ and $neighborhood(x_{2_k}, u)$ are similar; otherwise they are dissimilar. Such an assump-

tion derives from the consideration that two sets of objects can be considered similar if the number of similar components is greater than the number of the dissimilar ones or, in other words, if the number of similar components is greater than half of the total number of components.

We present now the following theorem stating the computational complexity of the x-components' similarity extraction.

Theorem 2. Let S_1 and S_2 be two XML documents. Let x_{1_j} (resp., x_{2_k}) be an x-component of S_1 (resp., S_2). Let u be the selected severity level. Finally, let p be the maximum between the cardinality of $neighborhood(x_{1_j}, u)$ and $neighborhood(x_{2_k}, u)$. The worst case time complexity for computing *synonymous*(x_{1_j}, x_{2_k}, u) is $O((u + 1) \times p^3)$. \square

Corollary 1. Let S_1 and S_2 be two XML documents. Let u be the severity level. Let m be the maximum between the number of complex elements of S_1 and S_2 . Finally, let q be the maximum cardinality relative to a neighborhood of S_1 or S_2 . The worst case time complexity for deriving all synonymies existing, at the severity level u , between S_1 and S_2 is $O((u + 1) \times q^3 \times m^2)$. \square

3.2 Derivation of homonymies

After synonymies among x-components of S_1 and S_2 have been extracted, homonymies can be directly derived from them. More specifically, we say that an homonymy holds between x_{1_j} and x_{2_k} with a severity level equal to u if *synonymous*(x_{1_j}, x_{2_k}, u) = *false* and both x_{1_j} and x_{2_k} have the same name.

It is possible to define a boolean function *homonymous*, which receives two x-components x_{1_j} and x_{2_k} and an integer u and returns *true* if there exists an homonymy between x_{1_j} and x_{2_k} with a severity level equal to u ; *homonymous* returns *false* otherwise.

Example 2. Consider the XML Schemas S_1 and S_2 , shown in Figures 1 and 2. Consider also the x-components $customer_{[S_1]}$ ² and $client_{[S_2]}$. In order to check if they are synonymous with a severity level 0, it is necessary to compute the function *synonymous*($customer_{[S_1]}, client_{[S_2]}, 0$). Now, *neighborhood*($customer_{[S_1]}, 0$) has been shown in Example 1; as for *neighborhood*($client_{[S_2]}, 0$), we have:

$$neighborhood(client_{[S_2]}, 0) = \{client_{[S_2]}, SSN_{[S_2]}, firstName_{[S_2]}, lastName_{[S_2]}, address_{[S_2]}, phone_{[S_2]}, email_{[S_2]}\}$$

The function $\phi_{BG}(0)$ computed by *synonymous* in this case is $\frac{2|A'(0)|}{|P(0)|+|Q(0)|} = \frac{2 \times 5}{8+7} = 0.67 > \frac{1}{2}$; therefore *synonymous*($customer_{[S_1]}, client_{[S_2]}, 0$) = *true*.

In an analogous way, *synonymous*($customer_{[S_1]}, client_{[S_2]}, 1$) can be computed. In particular, in this case, $\phi_{BG}(1) = 0.43 < \frac{1}{2}$; as a consequence,

² Here and in the following, we use the notation $x_{[S]}$ to indicate the x-component x of the XML Schema S .


```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Definition of attributes -->
  <xs:attribute name="SSN" type="xs:string"/>
  <xs:attribute name="code" type="xs:ID"/>
  <xs:attribute name="purchasedCDDAs" type="xs:IDREFS"/>
  <xs:attribute name="purchasedMiniDisks" type="xs:IDREFS"/>
  <xs:attribute name="purchaseDate" type="xs:date"/>
  <xs:attribute name="quantity" type="xs:integer"/>
  <xs:attribute name="bitRate" type="xs:integer"/>
  <!-- Definition of simple elements -->
  <xs:element name="firstName" type="xs:string"/>
  <xs:element name="lastName" type="xs:string"/>
  <xs:element name="address" type="xs:string"/>
  <xs:element name="phone" type="xs:string"/>
  <xs:element name="email" type="xs:string"/>
  <xs:element name="artist" type="xs:string"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="song" type="xs:string"/>
  <xs:element name="year" type="xs:integer"/>
  <xs:element name="genre" type="xs:string"/>
  <!-- Definition of complex elements -->
  <xs:element name="CDDAPurchase">
    <xs:complexType>
      <xs:sequence>
        <xs:attribute ref="purchaseDate"/>
        <xs:attribute ref="purchasedCDDAs"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="miniDiskPurchase">
    <xs:complexType>
      <xs:sequence>
        <xs:attribute ref="purchaseDate"/>
        <xs:attribute ref="purchasedMiniDisks"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="client">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="firstName"/>
        <xs:element ref="lastName"/>
        <xs:element ref="address"/>
        <xs:element ref="phone" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="email" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="CDDAPurchase" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element ref="miniDiskPurchase" minOccurs="0" maxOccurs="unbounded"/>
  </xs:schema>
  <xs:element ref="miniDiskPurchase" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="SSN" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="CDDA">
  <xs:complexType>
    <xs:attribute ref="code" use="required"/>
    <xs:attribute ref="quantity"/>
  </xs:complexType>
</xs:element>
<xs:element name="miniDisk">
  <xs:complexType>
    <xs:attribute ref="code" use="required"/>
    <xs:attribute ref="quantity"/>
    <xs:attribute ref="bitRate"/>
  </xs:complexType>
</xs:element>
<xs:element name="composition">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="artist" maxOccurs="unbounded"/>
      <xs:element ref="title"/>
      <xs:element ref="song" maxOccurs="unbounded"/>
      <xs:element ref="year"/>
      <xs:element ref="genre"/>
      <xs:element ref="CDDA" minOccurs="0"/>
      <xs:element ref="miniDisk" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- Definition of root element -->
<xs:element name="store">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="client" maxOccurs="unbounded"/>
      <xs:element ref="composition" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Fig. 2. The XML Schema S_2

$\text{synonymous}(\text{customer}_{[S_1]}, \text{client}_{[S_2]}, 1) = \text{false}$, i.e. $\text{customer}_{[S_1]}$ and $\text{client}_{[S_2]}$ cannot be considered synonymous with a severity level 1.

All the other synonymies can be derived analogously. As for these Schemas, no homonymy has been found. \square

4 The Integration Task

In this section we propose an integration algorithm which receives two XML Schemas S_1 and S_2 and a severity level u and returns the integrated XML Schema S_G . The algorithm consists of two steps, namely: (i) construction of a *Merge Dictionary* $MD(u)$ and a *Rename Dictionary* $RD(u)$; (ii) exploitation of $MD(u)$ and $RD(u)$ for obtaining the global Schema.

Preliminarily it is necessary to observe that in XML Schemas there exists a large variety of data types. Some of them, e.g. *Byte* and *Int*, are compatible in the sense that each attribute or simple element whose type is *Byte* can be treated as an attribute or a simple element whose type is *Int*; in this case *Int* is said *more general* than *Byte*. Other types, e.g. *Int* and *Date*, are not compatible. Compatibility rules are analogous to the corresponding ones valid for high level programming languages.

4.1 Construction of MD(u) and RD(u)

At the end of interscheme property derivation, it could happen that an x-component of a Schema is synonymous (resp., homonymous) with more than one x-components of the other Schema. The integration algorithm we are proposing here needs each x-component of a Schema to be synonymous (resp., homonymous) with at most one x-component of the other Schema. In order to satisfy this requirement, it is necessary to construct a *Merge Dictionary* $MD(u)$ and an *Rename Dictionary* $RD(u)$ by suitably filtering previously derived synonymies and homonymies.

The construction of $MD(u)$ begins with the definition of a support bipartite graph $SimG(u) = \langle SimNSet_1(u) \cup SimNSet_2(u), SimASet(u) \rangle$.

There is a node n_{1_j} (resp., n_{2_k}) in $SimNSet_1(u)$ (resp., $SimNSet_2(u)$) for each complex element E_{1_j} (resp., E_{2_k}) belonging to S_1 (resp., S_2). There is an arc $A_{jk} = \langle n_{1_j}, n_{2_k} \rangle \in SimASet(u)$ if $synonymous(E_{1_j}, E_{2_k}, u) = true$; the label of each arc A_{jk} is $f(n_{1_j}, n_{2_k})$ where:

$$f(n_{1_j}, n_{2_k}) = \begin{cases} \phi_{BG}(E_{1_j}, E_{2_k}, u) & \text{if } A_{jk} \in SimASet(u) \\ 0 & \text{otherwise} \end{cases}$$

Function f has been defined in such a way to maximize the sum of the similarity degrees involving complex elements of S_1 and S_2 .

After this, a maximum weight matching is computed on $SimG(u)$; this selects a subset $SimASubSet(u) \subseteq SimASet(u)$ which maximizes the objective function $\phi_{Sim}(u) = \sum_{\langle n_{1_j}, n_{2_k} \rangle \in SimASubSet(u)} f(n_{1_j}, n_{2_k})$.

For each arc $A'_{jk} = \langle n'_{1_j}, n'_{2_k} \rangle \in SimASubSet(u)$ a pair $\langle E'_{1_j}, E'_{2_k} \rangle$ is added to $MD(u)$.

In addition, let E'_{1_j} (resp., E'_{2_k}) be a complex element of S_1 (resp., S_2) such that $\langle E'_{1_j}, E'_{2_k} \rangle \in MD(u)$ and let x'_{1_j} (resp., x'_{2_k}) be an attribute or a simple element of E'_{1_j} (resp., E'_{2_k}); then $\langle x'_{1_j}, x'_{2_k} \rangle$ is added to $MD(u)$ if (i) a synonymy between the name of x'_{1_j} and that of x'_{2_k} holds in the reference thesaurus and the data types of x'_{1_j} and x'_{2_k} are compatible, or (ii) x'_{1_j} and x'_{2_k} have the same name, the same typology and compatible data types.

After $MD(u)$ has been constructed, it is possible to derive $RD(u)$. More specifically, a pair of x-components $\langle x''_{1_j}, x''_{2_k} \rangle$ is added to $RD(u)$ if x''_{1_j} and x''_{2_k} are two elements or two attributes having the same name and $\langle x''_{1_j}, x''_{2_k} \rangle \notin MD(u)$.

4.2 Construction of the global XML Schema

After $MD(u)$ and $RD(u)$ have been derived, it is possible to exploit them for constructing a global Schema S_G . Our integration algorithm assumes that S_1 and S_2 are represented in the *referenced style*, i.e., that they consist of sequences of elements and that each element may refer to other elements by means of the *ref* attribute. Actually, an XML Schema could be defined in various other ways (e.g., with the *inline style*); however, simple rules can be easily defined for translating it in the *referenced style* (see [1] for more details on the various definition styles).

More formally, S_1 and S_2 can be represented as:

$$S_1 = \langle x_{1_1}, x_{1_2}, \dots, x_{1_i}, \dots, x_{1_n} \rangle; S_2 = \langle x_{2_1}, x_{2_2}, \dots, x_{2_j}, \dots, x_{2_m} \rangle$$

where $x_{1_1}, \dots, x_{1_n}, x_{2_1}, \dots, x_{2_m}$ are x-components. A first, rough, version of S_G can be obtained by constructing a list containing all the x-components of S_1 and S_2 :

$$S_G = \langle x_{1_1}, \dots, x_{1_n}, x_{2_1}, \dots, x_{2_m} \rangle$$

This version of S_G could present some redundancies and/or ambiguities. In order to remove them and, consequently, to refine S_G , $MD(u)$ and $RD(u)$ must be examined and some tasks must be performed for each of the properties they store. More specifically, consider $MD(u)$ and let $\langle E_{1_j}, E_{2_k} \rangle \in MD(u)$ be a synonymy between two complex elements. E_{1_j} and E_{2_k} are merged into a complex element E_{jk} . The name of E_{jk} is one between the names of E_{1_j} and E_{2_k} . The set of sub-elements of E_{jk} is obtained by applying the **xs:sequence** indicator to the sets of sub-elements of E_{1_j} and E_{2_k} ; the list of attributes of E_{jk} is formed by the attributes of E_{1_j} and E_{2_k} . Note that, after these tasks have been carried out, it could happen that:

- A tuple $\langle A'_{jk}, A''_{jk} \rangle$, such that A'_{jk} and A''_{jk} are attributes of E_{jk} , belongs to $MD(u)$. In this case A'_{jk} and A''_{jk} are merged into an attribute A^*_{jk} ; the name of A^*_{jk} is one between the names of A'_{jk} and A''_{jk} ; the type of A^*_{jk} is the most general one between those of A'_{jk} and A''_{jk} .
- A tuple $\langle E'_{jk}, E''_{jk} \rangle$, such that E'_{jk} and E''_{jk} are simple elements of E_{jk} , belongs to $MD(u)$. In this case E'_{jk} and E''_{jk} are merged into an element E^*_{jk} ; the name of E^*_{jk} is one between the names of E'_{jk} and E''_{jk} ; the type of E^*_{jk} is the most general one between those of E'_{jk} and E''_{jk} ; the *minOccurs* (resp., the *maxOccurs*) indicator of E^*_{jk} is the minimum (resp., the maximum) between the corresponding ones relative to E'_{jk} and E''_{jk} .
- A tuple $\langle E''_{jk}, A''_{jk} \rangle$, such that E''_{jk} is a simple sub-element of E_{jk} and A''_{jk} is an attribute of E_{jk} , belongs to $MD(u)$. In this case, A''_{jk} is removed since its information content is equivalent to that of E''_{jk} and the representation of an information content by means of an element is more general than that obtained by exploiting an attribute.

After this, all references to E_{1_j} and E_{2_k} in S_G are transformed into references to E_{jk} ; the *maxOccurs* and the *minOccurs* indicators associated with E_{jk} are derived from the corresponding ones relative to E_{1_j} and E_{2_k} and, finally, E_{1_j} is replaced by E_{jk} whereas E_{2_k} is removed from S_G .

After $MD(u)$ has been examined, it is necessary to consider $RD(u)$; in particular, let $\langle x_{1_j}, x_{2_k} \rangle$ be a tuple of $RD(u)$ such that x_{1_j} and x_{2_k} are both elements or both attributes of the same element. In this case it is necessary to modify the name of either x_{1_j} or x_{2_k} and all the corresponding references.

Observe that, after all these activities have been performed, S_G could contain two root elements. Such a situation occurs when the root elements E_{1_r} of S_1 and E_{2_r} of S_2 are not synonymous. In this case it is necessary to create a new root element E_{G_r} in S_G whose set of sub-elements is obtained by applying the **xs:all** indicator to E_{1_r} and E_{2_r} . The occurrence indicators associated with E_{1_r} and E_{2_r} are *minOccurs* = 0 and *maxOccurs* = 1.

As for the computational complexity of the integration task, it is possible to state the following theorem.

<i>x-component of S_1</i>	<i>x-component of S_2</i>		<i>x-component of S_1</i>	<i>x-component of S_2</i>
shop	store		customer	client
music	composition		SSN	SSN
firstName	firstName		lastName	lastName
address	address		code	code
artist	artist		title	title
pubYear	year		genre	genre

Table 1. The Merge Dictionary $MD(0)$

Theorem 3. Let S_1 and S_2 be two XML Schemas, let n be the maximum between $|XCompSet(S_1)|$ and $|XCompSet(S_2)|$ and let m be the maximum between the number of complex elements of S_1 and the number of complex elements of S_2 . The worst case time complexity for integrating S_1 and S_2 into a global Schema S_G is $O(m \times n^2)$. \square

Example 3. Assume a user wants to integrate the XML Schemas S_1 and S_2 , shown in Figures 1 and 2, and the severity level she/he specifies is 0. $MD(0)$ is illustrated in Table 1; $RD(0)$ is empty because no homonymy has been found among x-components of S_1 and S_2 (see Example 2). Initially a rough version of S_G is constructed that contains all the x-components of S_1 and S_2 ; the refined version of S_G is obtained by removing (possible) redundancies and/or ambiguities present therein.

The first step of the refinement phase examines all synonymies among complex elements stored in $MD(0)$. As an example, consider the synonymous elements $customer_{[S_1]}$ and $client_{[S_2]}$; they must be merged in one single element. This task is carried out as follows. First a new element $customer_{[S_G]}$ is created in S_G . The set of sub-elements of $customer_{[S_G]}$ is obtained by applying the **xs:sequence** indicator to the sets of sub-elements of $customer_{[S_1]}$ and $client_{[S_2]}$; the list of attributes of $customer_{[S_G]}$ is formed by the attributes of $customer_{[S_1]}$ and $client_{[S_2]}$. At the end of this task, $customer_{[S_G]}$ contains two attributes named *SSN*. Since the tuple $\langle SSN, SSN \rangle$ belongs to $MD(0)$, the two attributes are merged into a single attribute *SSN* having type “string”. An analogous procedure is applied to sub-element pairs $\langle firstName_{[S_1]}, firstName_{[S_2]} \rangle$, $\langle lastName_{[S_1]}, lastName_{[S_2]} \rangle$ and $\langle address_{[S_1]}, address_{[S_2]} \rangle$.

After this, all references to $customer_{[S_1]}$ and $client_{[S_2]}$ are transformed into references to $customer_{[S_G]}$; finally, $customer_{[S_1]}$ is replaced by $customer_{[S_G]}$ whereas $client_{[S_2]}$ is removed from S_G . All the other synonymies stored in $MD(0)$ are handled similarly. Since no homonymy has been found, no further action is necessary. The global XML Schema S_G , obtained at the end of the integration activity, is shown in Figure 3. \square

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Definition of attributes -->
  <xs:attribute name="SSN" type="xs:string"/>
  <xs:attribute name="code" type="xs:ID"/>
  <xs:attribute name="acquiredBooks" type="xs:IDREFS"/>
  <xs:attribute name="acquiredMusics" type="xs:IDREFS"/>
  <xs:attribute name="acquirementDate" type="xs:date"/>
  <xs:attribute name="purchasedCDDAs" type="xs:IDREFS"/>
  <xs:attribute name="purchasedMiniDisks" type="xs:IDREFS"/>
  <xs:attribute name="purchaseDate" type="xs:date"/>
  <xs:attribute name="quantity" type="xs:integer"/>
  <xs:attribute name="bitRate" type="xs:integer"/>
  <!-- Definition of simple elements -->
  <xs:element name="firstName" type="xs:string"/>
  <xs:element name="lastName" type="xs:string"/>
  <xs:element name="address" type="xs:string"/>
  <xs:element name="gender" type="xs:string"/>
  <xs:element name="birthDate" type="xs:date"/>
  <xs:element name="profession" type="xs:string"/>
  <xs:element name="phone" type="xs:string"/>
  <xs:element name="email" type="xs:string"/>
  <xs:element name="artist" type="xs:string"/>
  <xs:element name="author" type="xs:string"/>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="song" type="xs:string"/>
  <xs:element name="pubYear" type="xs:integer"/>
  <xs:element name="publisher" type="xs:string"/>
  <xs:element name="genre" type="xs:string"/>
  <xs:element name="support" type="xs:string"/>
  <!-- Definition of complex elements -->
  <xs:element name="bookAcquirement">
    <xs:complexType>
      <xs:attribute ref="acquirementDate"/>
      <xs:attribute ref="acquiredBooks"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="musicAcquirement">
    <xs:complexType>
      <xs:attribute ref="acquirementDate"/>
      <xs:attribute ref="acquiredMusics"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="author" maxOccurs="unbounded"/>
        <xs:element ref="title"/>
        <xs:element ref="publisher"/>
        <xs:element ref="pubYear"/>
        <xs:element ref="genre"/>
      </xs:sequence>
      <xs:attribute ref="code" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="CDDAPurchase">
    <xs:complexType>
      <xs:attribute ref="purchaseDate"/>
      <xs:attribute ref="purchasedCDDAs"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="miniDiskPurchase">
    <xs:complexType>
      <xs:attribute ref="purchaseDate"/>
      <xs:attribute ref="purchasedMiniDisks"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="firstName"/>
        <xs:element ref="lastName"/>
        <xs:element ref="address"/>
        <xs:element ref="gender"/>
        <xs:element ref="birthDate"/>
        <xs:element ref="profession"/>
        <xs:element ref="bookAcquirement"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="musicAcquirement"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="phone"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="email"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="CDDAPurchase"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="miniDiskPurchase"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute ref="SSN" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="CDDA">
    <xs:complexType>
      <xs:attribute ref="code"/>
      <xs:attribute ref="quantity"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="miniDisk">
    <xs:complexType>
      <xs:attribute ref="code"/>
      <xs:attribute ref="quantity"/>
      <xs:attribute ref="bitRate"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="music">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="artist" maxOccurs="unbounded"/>
        <xs:element ref="title"/>
        <xs:element ref="pubYear"/>
        <xs:element ref="genre"/>
        <xs:element ref="support" minOccurs="0"/>
        <xs:element ref="song"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="CDDA" minOccurs="0"/>
        <xs:element ref="miniDisk" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute ref="code"/>
    </xs:complexType>
  </xs:element>
  <!-- Definition of root element -->
  <xs:element name="shop">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="customer"
          maxOccurs="unbounded"/>
        <xs:element ref="music"
          maxOccurs="unbounded"/>
        <xs:element ref="book"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Fig. 3. The integrated XML Schema S_G

5 Experiments

To test the performances of our approach we have carried out various experiments; these have been performed on several XML Schemas taken from different application contexts. Involved XML Schemas were very heterogeneous in their dimensions; indeed, the number of x-components associated with them ranged from tens to hundreds.

The first series of experiments has been conceived for measuring correctness and completeness of our interscheme property derivation algorithm. In particular, *correctness* lists the percentage of properties returned by our techniques agreeing with those provided by humans; *completeness* lists the percentage of properties returned by our approach with regard to the set of properties provided by humans.

In more detail, we proceeded as follows: *(i)* we ran our algorithms on several pairs of XML Schemas and collected the returned results; *(ii)* for each pair of Schemas we asked humans to specify a set of significant interscheme properties; *(iii)* we computed the overall quality figures by comparing the set of properties obtained as described at points 1 and 2 above.

As for severity level 0, we have obtained a correctness equal to 0.88 and a completeness equal to 1.00.

Actually, the intrinsic characteristics of our algorithm led us to think that, if the severity level increases, the correctness increases as well, whereas the completeness decreases. In order to verify this idea, we have performed a second series of experiments devoted to measure correctness and completeness in presence of variations of the severity level. Table 2 shows obtained results up to a severity level equal to 3; for higher severity levels, variations of correctness and completeness are not significant.

<i>Severity Level</i>	<i>Correctness</i>	<i>Completeness</i>
Level 0	0.88	1.00
Level 1	0.97	0.81
Level 2	0.97	0.78
Level 3	0.97	0.73

Table 2. Correctness and Completeness of our approach at various severity levels

Results presented in Table 2 confirmed our intuitions. Indeed, at severity level 1, correctness increases of a factor of 9% whereas completeness decreases of a factor of 19% w.r.t. correctness and completeness relative to severity level 0. As for severity levels greater than 1, we have verified that correctness does not increase whereas completeness slightly decreases w.r.t. level 1.

In our opinion such a result is extremely relevant; indeed, it allows us to conclude that, in informal situations, the right severity level is 0 whereas, in more formal contexts, the severity level must be at least 1.

After this, we have computed variations of the time required for deriving interscheme properties caused by an increase of the severity level. Obtained results are shown in Table 3. In the table the value associated with severity level i ($1 \leq i \leq 3$) is to be intended as the percentage of time additionally required w.r.t. severity level $i - 1$.

<i>Severity Level</i>	<i>Time Increase</i>
Level 1	56%
Level 2	14%
Level 3	20%

Table 3. Increase of the time required by our approach at various severity levels

Table 3 shows that the increase of time required for computing interscheme properties when the algorithm passes from the severity level 0 to the severity level 1 is significant. Vice versa, further severity level increases do not lead to significant increases of the time necessary for computing interscheme properties. This observation further confirms results obtained by the previous experiments, i.e., that the most relevant differences in the results obtained by applying our approach can be found between the severity levels 0 and 1.

6 Related Work

In the literature many approaches for performing interscheme property extraction and data source integration have been proposed. Even if they are quite numerous and various, to the best of our knowledge, none of them guarantees the possibility to choose a “severity” level against which the various activities are carried out. In this section we examine some of these approaches and highlight their similarities and differences w.r.t. our own.

In [16] an XML Schema integration framework is proposed. It consists of three phases, namely pre-integration, comparison and integration. After this, *conflict resolution* and *restructuring* are performed for obtaining the global refined Schema. To the best of our knowledge the approach of [16] is the closest to our own. In particular, (i) both of them are *rule-based* [17]; (ii) both of them assume that the global Schema is formulated in a *referenced style* rather than in an *inline style* (see [1] for more details); (iii) integration rules proposed in [16] are quite similar to those characterizing our approach. The main differences existing between them are the following: (i) the approach of [16] requires a preliminary translation of an XML Schema into an *XSDM* Schema; such a translation is not required by our approach; (ii) the integration task in [16] is graph-based and object-oriented whereas, in our approach, it is directly based on *x-components*;

In [10] the system *XClust* is presented whose purpose is XML data source integration. More specifically, *XClust* determines the similarity degrees of a group of DTD’s by considering not only the corresponding linguistic and structural

information but also their semantics. It is possible to recognize some similarities between our approach and *XClust*; in particular, (i) both of them have been specifically conceived for operating on XML data sources (even if our approach manages XML Schemas whereas *XClust* operates on DTD's); (ii) both of them consider not only linguistic similarities but also semantic ones. There are also several differences between the two approaches; more specifically, (i) to perform the integration activity, *XClust* requires the support of a hierarchical clustering whereas our approach adopts schema matching techniques; (ii) *XClust* represents DTD's as trees; as a consequence, element neighborhoods are quite different from those constructed by our approach; (iii) *XClust* exploits some weights and thresholds whereas our approach does not use them; as a consequence, *XClust* provides more refined results but these last are strongly dependent on the correctness of a tuning phase devoted to set weights and thresholds.

In [13] the system *Rondo* is presented. It has been conceived for integrating and manipulating relational schemas, XML Schemas and SQL views. *Rondo* exploits a graph-based approach for modeling information sources and a set of high-level operators for matching obtained graphs. *Rondo* uses the *Similarity Flooding Algorithm*, a graph-matching algorithm proposed in [12], to perform schema matching activity. Finally, it merges involved information sources according to three steps: Node Renaming, Graph Union and Conflict Resolution. There are important similarities between *Rondo* and our approach; indeed both of them are semi-automatic and exploit schema matching techniques. The main differences existing between them are the following: (i) *Rondo* is generic, i.e., it can handle various kinds of information sources; vice versa our approach is specialized for XML Schemas; (ii) *Rondo* models involved information sources as graphs whereas our approach directly operates on XML Schemas; (iii) *Rondo* exploits a sophisticated technique (i.e., the Similarity Flooding Algorithm) for carrying out schema matching activities [12]; as a consequence, it obtains very precise results but is time-expensive and requires a heavy human feedback; on the contrary, our approach is less sophisticated but is well suited when involved information sources are numerous and large.

In [6] an XML-based integration approach, capable of handling various source formats, is presented. Both this approach and our own operate on XML documents and carry out a semantic integration. However, (i) the approach of [6] operates on DTD's and requires to translate them in an appropriate formalism called ORM/NIAM [9]; vice versa, our approach directly operates on XML Schemas; (ii) the global Schema constructed by the approach of [6] is represented in the ORM/NIAM formalism whereas our approach directly returns a global XML Schema; (iii) the approach of [6] is quite complex to be applied when involved sources are numerous.

In [18] the DIXSE (Data Integration for XML based on Schematic Knowledge) tool is presented, aiming at supporting the integration of a set of XML documents. Both DIXSE and our approach are semantic and operate on XML documents; both of them exploit structural and terminological relationships for carrying out the integration activity. The main differences between them reside

in the interscheme property extraction technique; indeed, DIXSE requires the support of the user whereas our approach derives them almost automatically. As a consequence, results returned by DIXSE could be more precise than those provided by our approach but, when the number of sources to integrate is high, the effort DIXSE requires to the user might be particularly heavy.

In [4] a *machine learning* approach, named LSD (Learning Source Description), for carrying out schema matching activities, is proposed. It has been extended also to ontologies in GLUE [5]. LSD requires quite a heavy support of the user during the initial phase, for carrying out training tasks; however, after this phase, no human intervention is required. Both LSD and our approach operate mainly on XML sources. They differ especially in their purposes; indeed, LSD aims at deriving interscheme properties whereas our approach has been conceived mainly for handling integration activities. In addition, as far as interscheme property derivation is concerned, it is worth observing that LSD is “*learner-based*” whereas our approach is “*rule-based*” [17]. Finally, LSD requires a heavy human intervention at the beginning and, then, is automatic; vice versa, our approach does not need a pre-processing phase but requires the human intervention at the end for validating obtained results.

In [3] the authors propose COMA (COMbining MATch), an interactive and iterative system for combining various schema matching approaches. The approach of COMA appears orthogonal to our own; in particular, our approach could inherit some features from COMA (as an example, the idea of operating iteratively) for improving the accuracy of its results. As for an important difference between the two approaches, we observe that COMA is generic, since it handles a large variety of information source formats; vice versa, our approach has been specifically conceived to handle XML documents. In addition, our approach requires the user to specify only the *severity level*; vice versa, in COMA, the user must specify the *matching strategy* (i.e., the desired matchers to exploit and the modalities for combining their results).

7 Conclusions

In this paper we have proposed an approach for the integration of a set of XML Schemas. We have shown that our approach is specialized for XML documents, is almost automatic, semantic and “light” and allows the choice of the “severity” level against which the integration activity must be performed. We have also illustrated some experiments we have carried out to test its computational performances and the quality of results it obtains. Finally, we have examined various other related approaches previously proposed in the literature and we have compared them with ours by pointing out similarities and differences.

In the future we plan to exploit our approach in various other contexts typically benefiting of information source integration, such as Cooperative Information Systems, Data Warehousing, Semantic Query Processing and so on.

References

1. XML Schema Part 1: Structures. W3C Recommendation, <http://www.w3.org/TR/xmlschema-1>, 2001.
2. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2):277–297, 2001.
3. H. Do and E. Rahm. COMA- a system for flexible combination of schema matching approaches. In *Proc. of the International Conference on Very Large Databases (VLDB 2002)*, pages 610–621, Hong Kong, China, 2002. VLDB Endowment.
4. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. of the ACM International Conference on Management of Data (SIGMOD 2001)*, pages 509–520, Santa Barbara, California, USA, 2001. ACM Press.
5. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *Proc. of the ACM International Conference on World Wide Web (WWW 2002)*, pages 662–673, Honolulu, Hawaii, USA, 2002. ACM Press.
6. R. dos Santos Mello, S. Castano, and C.A. Heuser. A method for the unification of XML schemata. *Information & Software Technology*, 44(4):241–249, 2002.
7. P. Fankhauser, M. Kracker, and E.J. Neuhold. Semantic vs. structural resemblance of classes. *ACM SIGMOD RECORD*, 20(4):59–63, 1991.
8. Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18:23–38, 1986.
9. T. Halpin. Object-Role Modeling (ORM-NIAM). In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on Architectures of Information Systems*, chapter 4, pages 81–102. Springer-Verlag, 1998.
10. M.L. Lee, L.H. Yang, W. Hsu, and X. Yang. XClust: clustering XML schemas for effective integration. In *Proc. of the ACM International Conference on Information and Knowledge Management (CIKM 2002)*, pages 292–299, McLean, Virginia, USA, 2002. ACM Press.
11. J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. of the International Conference on Very Large Data Bases (VLDB 2001)*, pages 49–58, Roma, Italy, 2001. Morgan Kaufmann.
12. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A versatile graph matching algorithm and its application to schema matching. In *Proc. of the IEEE International Conference on Data Engineering (ICDE 2002)*, pages 117–128, San Jose, California, USA, 2002. IEEE Computer Society Press.
13. S. Melnik, E. Rahm, and P.A. Bernstein. Rondo: A programming platform for generic model management. In *Proc. of the International Conference on Management of Data (SIGMOD 2003)*, pages 193–204, San Diego, California, USA, 2003. ACM Press.
14. A.G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
15. L. Palopoli, D. Saccà, G. Terracina, and D. Ursino. Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):271–294, 2003.
16. K. Passi, L. Lane, S.K. Madria, B.C. Sakamuri, M.K. Mohania, and S.S. Bhowmick. A model for XML Schema integration. In *Proc. of the International Conference on E-Commerce and Web Technologies (EC-Web 2002)*, pages 193–202, Aix-en-Provence, France, 2002. Lecture Notes in Computer Science, Springer.

17. E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
18. P. Rodriguez-Gianolli and J. Mylopoulos. A semantic approach to XML-based data integration. In *Proc. of the International Conference on Conceptual Modelling (ER'01)*, pages 117–132, Yokohama, Japan, 2001. Lecture Notes in Computer Science, Springer.