Journal of Universal Computer Science, vol. 10, no. 9 (2004), 1065-1109
submitted: 4/11/03, accepted: 19/5/04, appeared: 28/9/04

X-Global: a System for the "Almost Automatic" and Semantic Integration of XML Sources at Various Flexibility Levels

Pasquale De Meo
(DIMET, Università "Mediterranea" di Reggio Calabria
Via Graziella, Località Feo di Vito, 89060 Reggio Calabria, Italy
demeo@ing.unirc.it)

Giorgio Terracina
(Dipartimento di Matematica, Università della Calabria
Via Pietro Bucci, 87036 Arcavacata di Rende (CS), Italy
terracina@mat.unical.it)

Domenico Ursino
(DIMET, Università "Mediterranea" di Reggio Calabria
Via Graziella, Località Feo di Vito, 89060 Reggio Calabria, Italy
ursino@unirc.it)

Abstract: In this paper we propose *X-Global*, a novel, almost automatic and semantic system for integrating a set of XML sources. *X-Global* is parametric w.r.t. the flexibility level against which the integration task is performed; indeed, it can operate on rigid contexts, when two concepts are merged only if they have exactly the same meaning, as well as on flexible and informal situations, when two concepts are merged if they have close, even if not exactly identical, meanings. In this paper we describe the system in all details, illustrate various theoretical results as well as several experiments we have carried out for verifying its performance. Finally, we examine related literature and point out similarities and differences between *X-Global* and several other approaches already proposed in the past.
Key Words: Information Source Integration, Interschema Property, XML
Category: H.2.4, H.3.4

1 Introduction

1.1 Motivations

The Web is presently becoming the most important infrastructure for both the publication and the exchange of information among various organizations. Its rapidly increasing and permeating diffusion, along with the easiness on accessing and exploiting it, allows us to foresee that it will play a key role in information management in the near future.

In order to support such a role, XML (eXtensible Markup Language) has been proposed by the W3C (World Wide Web Consortium) as the standard language for both representing and exchanging information on the Web. Such a proposal is having a wide success and most of the organizations operating on the Web are adopting, or will adopt, XML for both handling and disseminating their information.

Certainly, XML plays a key role in the support of the interoperability of Web information sources; however, it is not enough to fulfill such a task. As a matter of fact, the heterogeneity of data exchanged over the Web regards not only their formats but also their semantics. XML allows to face the format heterogeneities; in order to successfully handle the semantic ones, an integration task is necessary.

This paper aims at providing a contribution in this setting; in particular, it presents *X-Global*, a system specifically conceived for XML source integration.

1.2 General characteristics of the approach

X-Global is characterized by the following features: *(i)* it is specialized for XML documents; *(ii)* it is almost automatic; *(iii)* it is semantic; *(iv)* it allows the choice of the flexibility degree against which the integration task is performed. In the following, we shall discuss these features in more detail.

– *X-Global is specialized for XML documents.* In the literature, various methodologies have been proposed for integrating information sources having different formats and structure degrees (e.g., databases, XML documents, OEM graphs, object-oriented sources and so on). Generally, they translate all involved information sources into a common representation (such as a hierarchical or a graph-based or an object-oriented one) and, then, carry out the integration activity. Other systems have been proposed for operating only on a specific kind of sources, e.g., databases or XML documents. *X-Global* belongs to this latter group since it assumes all involved sources to be XML documents. With regard to this, it is worth observing that: *(i)* the integration of XML documents will play a key role in the future; *(ii)* the exploitation of generic approaches, capable of operating on information sources with different formats, for performing the integration of a set of XML sources (i.e., a set of sources having the same format), is unnecessarily expensive and inefficient. Indeed, it would require the translation of involved XML sources in another format as well as the translation of the integrated source from such a format back into XML.

– *X-Global is almost automatic.* Owing to the enormous increase of the number of available information sources, all integration techniques 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 implies a further effort in conceiving more automatic approaches. Moreover, it is worth observing 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 and complex. *X-Global* is almost automatic and simple to be applied. It does not exploit any threshold or weight; therefore, the user intervention is required only for determining the flexibility degree (see below) and for validating the obtained results. The lack of thresholds and weights makes *X-Global* simple and "light" since it does not need a tuning activity.

– *X-Global is semantic.* In the literature, various studies proved that the semantics of concepts belonging to involved information sources needs to be considered during the integration task [3, 8, 11, 22]. Given two concepts c_1, belonging to a source S_1, and c_2, belonging to a source S_2, one of the most common ways for determining their semantics consists in examining the concepts someway related to them in S_1 and S_2 since these concepts contribute to define the contexts which they have been defined in, and, consequently, to precisely define their meaning. As an example, suppose S_1 is an information source relative to vegetables and S_2 concerns factories and assume that the concept "plant" is present in both of them. If we examine the two concepts "plant" alone i.e., without considering their corresponding contexts, we could erroneously conclude that they are equal since they have the same name. Vice versa, if we examine both the two concepts "plant" and the concepts someway related to them, we can easily determine that these two concepts are completely different and represent different meanings. We call neighborhood of a concept c the set of concepts someway related to it in the corresponding source[1]. This reasoning leads us to conclude that two concepts, belonging to different information sources, are considered semantically similar (and are merged in the integrated source) if their neighborhoods are similar, and that they are considered semantically different (and are not merged in the integrated source) if their neighborhoods are dissimilar. In such a context, the closer to the concepts into examination the neighborhoods are, the stronger their relevance and contribution in determining if the concepts are similar or dissimilar will be. *X-Global* follows exactly such a philosophy.

– *X-Global allows the choice of the flexibility degree against which the inte-*

[1] As an example, the neighborhood of an element in an XML source could consist of the set of its sub-elements, its attributes and those elements whose instances are connected to it by its IDREF(S) attributes.

gration task must be performed. This is one of the most innovative features of our system. It 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 low flexibility degree is allowed. In other situations (e.g., tourist Web pages) the integration task can be looser and the user can decide to merge two concepts having some similarities but presenting also some differences.

At the beginning of the integration activity *X-Global* asks the user to specify the desired flexibility degree by means of a suitable, user-oriented interface; this is the only information requested 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 other approach handling the information source integration at various flexibility levels has 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 flexibility level is fixed and all concept merges are performed w.r.t. this level.

X-Global behaves as follows: first it determines the neighborhoods of the elements and the attributes of the XML sources to integrate; these neighborhoods are, then, used for computing interscheme properties (i.e., terminological and structural relationships) holding among attributes and elements belonging to involved XML sources [3, 11, 22, 30, 28, 29]. After this, some of the obtained properties are exploited for modifying involved sources in order to make them structurally and semantically uniform. The modified sources are, finally, integrated for obtaining the global source.

1.3 Plan of the paper

The plan of the paper is as follows: Section 2 is devoted to introduce some preliminary definitions; the neighborhood construction is the argument of Section 3. Section 4 illustrates our technique for extracting interscheme properties; the algorithm for constructing the global source is presented in Section 5. Section 6 illustrates the results of the experiments we have performed for testing our approach; the description of related work, along with the analysis of their similarities and differences w.r.t. our approach, is the argument of Section 7. Finally, in Section 8, we draw our conclusions.

2 Preliminaries

In this section we introduce some concepts that are widely used throughout the paper. Here, we assume the availability of the DTD of the documents into consideration; however, it is not strictly needed. Indeed, if an XML document does not have an associated DTD, it is possible to construct a valid one for it by applying one of the approaches already proposed in the literature for carrying out such a task [13, 27]. As for the computational complexity of the derivation of a DTD from an XML document we observe that this problem can be viewed as a Facility Location Problem (FLP), which is known to be NP-complete [15]; however, suitable heuristics have been proposed in the past for solving it. We adopt the heuristics described in [13] whose computational cost is $O(N_{inst}^2 \times log(N_{inst}))$, where N_{inst} is the number of instances of the XML document.

First we introduce the concept of x-component that allows both elements and attributes composing an XML document to be uniformly handled.

Definition 1. Let D be an XML document; an *x-component* of D is an element or an attribute of D. □

An x-component is characterized by its *name*, its *type*, its *cardinality* and, if it is an element, by its *content specification*. These last three features are better specified by the following definitions.

Definition 2. Let x_D be an x-component of a document D. The *type* of x_D indicates if it is an element or an attribute. In this latter case, the type of x_D indicates also if it is a CDATA, an ID, an IDREF, an IDREFS, a NMTOKEN, a NMTOKENS, an ENTITY, an ENTITIES, a NOTATION or an ENUMERATED attribute. □

Definition 3. Let x_D be a sub-element or an attribute of an element x'_D in an XML document D. The *minimum* (resp., *maximum*) *cardinality* of x_D w.r.t. x'_D indicates the minimum (resp., maximum) number of instances of x_D that can be associated with one instance of x'_D. □

As an example, if x_D is a sub-element of x'_D associated with the symbol '?', its minimum cardinality is 0 whereas its maximum cardinality is 1. If x_D is an #IMPLIED attribute of x'_D, its minimum cardinality is 0 and its maximum cardinality is 1. The cardinalities for all the other cases can be determined analogously.

Definition 4. Let x_D be an element of an XML document D. The *content specification* of x_D indicates the typology of the information it contains. The possible cases are:

- ANY, indicating that x_D may contain any information typology;
- EMPTY, denoting that no information typology can be associated with x_D;

– #PCDATA, denoting that the typology of the information contained in x_D is a string;

– a set of sub-elements $(x_{D_1}, x_{D_2}, \ldots, x_{D_n})$, indicating that the information content of x_D is specified by the information contents of $x_{D_1}, x_{D_2}, \ldots, x_{D_n}$;

– the OR of some content specifications (denoted by the symbol "|"), indicating that the information content of x_D could be one of them. $\Box$

This definition implicitly introduces a model for specifying the typology of the information associated with an element of an XML document D. This model simplifies the standard definition of element information typology in such a way to make the description of our integration approach easier. However, as it will be clear in the following, this simplification is not a limitation since our integration algorithm has been defined in such a way to guarantee that all the elements in the integrated document conform to the standard model for element information typology.

Finally, we introduce the following definitions that will be exploited frequently in the next sections.

Definition 5. Let D be an XML document; the set of its x-components is denoted as $XCompSet(D)$. $\Box$

Definition 6. Let x_j and x_k be two x-components of a document D. We say that a *first kind type incompatibility* exists between x_j and x_k if x_j is an element (resp., an attribute) and x_k is an attribute (resp., an element). $\Box$

Definition 7. Let x_j and x_k be two attributes of an XML document. We say that a *type heterogeneity* exists between x_j and x_k if they have different types (e.g., x_j is a CDATA attribute and x_k is an IDREF attribute). $\Box$

Consider now the attribute types of an XML document, as specified in Definition 2. As far as the integration task is concerned, we can observe that some of them are compatible, i.e. the corresponding attributes, if semantically similar, could be merged in the integrated document, whereas some of them are incompatible. In order to formalize this concept, we have defined a *Compatibility Matrix* M_C; it has 10 rows and 10 columns, one for each attribute type. In particular, $M_C[i,j] =$ *true* if i and j are compatible, $false$ otherwise. M_C is quite a sparse matrix. In Table 1 we show all pairs for which $M_C[i,j]$ is true. For all the other pairs $M_C[i,j]$ is false.

In the definition of M_C we have taken into account the meaning of the various attribute types in XML and the consequences on the semantics of the global document arising if attributes of different types would be integrated.

Attribute Type	*Attribute Type*		*Attribute Type*	*Attribute Type*
CDATA	CDATA		CDATA	NMTOKEN
CDATA	NMTOKENS		ID	ID
IDREF	IDREF		IDREF	IDREFS
IDREFS	IDREFS		NMTOKEN	NMTOKEN
NMTOKEN	NMTOKENS		NMTOKENS	NMTOKENS
NOTATION	NOTATION		NOTATION	ENTITY
NOTATION	ENTITIES		ENTITY	ENTITY
ENTITY	ENTITIES		ENTITIES	ENTITIES

Table 1: Definition of the Compatibility Matrix M_C: pairs for which $M_C[i,j] = true$

As an example, if we merge an IDREF and a CDATA attribute, the resulting one can be neither CDATA nor IDREF because, in both cases, the mapping with one of the original attributes would be impossible; as a consequence, $M_C[CDATA, IDREF] = false$. Vice versa, if we merge an IDREF and an IDREFS attribute, and the global attribute is set as IDREFS, both mappings between the integrated attribute and the original ones are possible; therefore, $M_C[IDREF, IDREFS] = true$.

The introduction of M_C allows the definition of the concept of second kind type incompatibility. It is specified by the following definition:

Definition 8. Let x_j and x_k be two XML attributes and let T_j (resp., T_k) be the type of x_j (resp., x_k). We say that a *second kind type incompatibility* exists between x_j and x_k if $M_C[T_j, T_k] = false$. $\Box$

Example 1. Consider the XML document U_1 shown in Figure 1, representing a University, and the corresponding DTD, shown in Figure 2.

Here *Professor* is an x-component because it is an element of U_1; analogously, *Name* is an x-component because it is an attribute of *Professor*. The *type* of the attribute *Salary* is CDATA because, in the DTD of U_1, it is declared as CDATA; analogously, the type of the attribute *Teaches_in* is IDREFS. The minimum cardinality of the sub-element *Phone* of the element *Professor* is one because, in the DTD, *Phone* is associated with the symbol '+'; analogously, the minimum cardinality of the attribute *Name* of the element *Professor* is one since, in the DTD, it is declared as #REQUIRED. Finally:[2]

$$\begin{aligned} XCompSet(U_1) = \{ & University, Professor, Identifier_{\langle Professor\rangle},\\ & Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle}, Salary_{\langle Professor\rangle},\\ & Belongs\_to_{\langle Professor\rangle}, Teaches\_in_{\langle Professor\rangle}, Phone, e\text{-}mail,\\ & Department, Identifier_{\langle Department\rangle}, Director_{\langle Department\rangle},\\ & Floor_{\langle Department\rangle}, Course, Identifier_{\langle Course\rangle}, Name_{\langle Course\rangle},\\ & Student\_Number_{\langle Course\rangle}, Argument_{\langle Course\rangle}, Duration_{\langle Course\rangle}, \end{aligned}$$

[2] Here and in the following, when the name of an attribute is ambiguous for determining the element it is associated with, we use the notation $x_{A_{\langle x_E\rangle}}$ to indicate that x_A is an attribute of the element x_E.

```
<?xml version=‘‘1.0’’?>
<!DOCTYPE university SYSTEM "university.dtd">
<University>
    <Professor Identifier="P156" Name="J.Sutt"
          Birthdate="15/06/50" Salary="2000"
          Belongs_to="D1" Teaches_in="C156" >
       <Phone> 336677 </Phone>
       <e-mail> J.Sutt@hotmail.com </e-mail>
       <e-mail> Sutt@di.uk </e-mail>
    </Professor>
    <Professor Identifier="P151" Name="K.Duke"
          Birthdate="13/05/48" Salary="2500"
          Belongs_to="D1" Teaches_in="C633" >
       <Phone> 336656 </Phone>
       <e-mail> Duke@di.uk </e-mail>
    </Professor>
    <Professor Identifier="P152" Name="B.Scott"
          Birthdate="13/03/52" Salary="2000">
       <Phone> 336674 </Phone>
       <Phone> 336671 </Phone>
       <e-mail> Scott@di.uk </e-mail>
    </Professor>
    <Department Identifier="D1" Director="A. Casey"
          Floor="3">
    </Department>
    <Course Identifier="C156" Name="Physics1"
          Student_Number="21" Argument="Physics"
          Duration="4" Attended_by="S01 S13"
          Teached_by="P156">
    </Course>
    <Course Identifier="C633" Name="Physics2"
          Student_Number="15" Argument="Physics"
          Duration="4" Attended_by="S10 S13"
          Teached_by="P151">
    </Course>
    <Student Identifier="S01" Name="D.Berg"
          Average_Mark="28.4" Birthdate="3/10/1976"
          Enrollment_Year="1995" Attends="C156">
    </Student>
    <Student Identifier="S10" Name="L.Marsh"
          Average_Mark="28.9" Birthdate="22/05/1978"
          Enrollment_Year="1996" Attends="C633">
       <Test Identifier="T09" Date="21/12/97"
             Mark="29" Argument="Chemistry">
       </Test>
       <Test Identifier="T15" Date="23/11/97"
             Mark="27" Argument="Physics">
       </Test>
    </Student>
    <Student Identifier="S13" Name="A.Bean"
          Average_Mark="27.3" Birthdate="12/07/1977"
          Enrollment_Year="1996" Tutor="P.Owen">
       <Test Identifier="T01" Date="23/11/97"
             Mark="21" Argument="Physics">
       </Test>
    </Student>
</University>
```

Figure 1: The XML document U_1 representing a University

```
<!ELEMENT University (Professor+,Department+,Student+)>
<!ELEMENT Professor (Phone+,e-mail*)>
    <!ATTLIST Professor
       Identifier ID #REQUIRED
       Name CDATA #REQUIRED
       Birthdate CDATA #IMPLIED
       Salary CDATA #IMPLIED
       Belongs_to IDREF #IMPLIED
       Teaches_in IDREFS #IMPLIED
    >
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
<!ELEMENT Department (EMPTY)>
    <!ATTLIST Department
       Identifier ID #REQUIRED
       Director CDATA #REQUIRED
       Floor CDATA #IMPLIED
    >
<!ELEMENT Course (EMPTY)>
    <!ATTLIST Course
       Identifier ID #REQUIRED
       Name CDATA #REQUIRED
       Student_Number CDATA #IMPLIED
       Argument CDATA #IMPLIED
       Duration CDATA #REQUIRED
       Attended_by IDREFS #IMPLIED
       Teached_by IDREFS #REQUIRED
    >
<!ELEMENT Student (Test*)>
    <!ATTLIST Student
       Identifier ID #REQUIRED
       Name CDATA #REQUIRED
       Average_Mark CDATA #REQUIRED
       Birthdate CDATA #REQUIRED
       Enrollment_Year CDATA #REQUIRED
       Tutor CDATA #IMPLIED
       Attends IDREFS #IMPLIED
    >
<!ELEMENT Test>
    <!ATTLIST Test
       Identifier ID #REQUIRED
       Date CDATA #REQUIRED
       Mark CDATA #REQUIRED
       Argument CDATA #REQUIRED
    >
```

Figure 2: The DTD of the XML document U_1

$$
\begin{aligned}
&Attended\_by_{\langle Course\rangle}, Teached\_by_{\langle Course\rangle}, Student, Identifier_{\langle Student\rangle},\\
&Name_{\langle Student\rangle}, Average\_Mark_{\langle Student\rangle}, Birthdate_{\langle Student\rangle},\\
&Enrollment\_Year_{\langle Student\rangle}, Tutor_{\langle Student\rangle}, Attends_{\langle Student\rangle}, Test,\\
&Identifier_{\langle Test\rangle}, Date_{\langle Test\rangle}, Mark_{\langle Test\rangle}, Argument_{\langle Test\rangle}\}
\end{aligned}
$$

□

3 Construction of x-components' neighborhoods

In this section we formally introduce the concept of neighborhood of an x-component. As pointed out in the Introduction, such a concept plays a key role in *X-Global*.

Given an x-component, in order to determine its neighborhoods, it is necessary to compute a sort of connection cost between it and each of the other x-components of the same XML document. The computation of such a connection cost requires to determine the strength of the relationship existing between two x-components or, in other words, when they can be considered semantically close and when, vice versa, they are to be considered semantically distant. In order to formalize these ideas, it is necessary to introduce the following functions:

- $veryclose(x_S, x_T)$, that returns *true* if and only if x_S is defined as element in the DTD of D and x_T is an attribute of x_S;
- $close(x_S, x_T)$, that returns *true* if and only if: *(i)* x_S is defined as element in the DTD of D and x_T is defined as sub-element in the DTD of x_S, or *(ii)* x_T is defined as element in the DTD of D and x_S has an IDREF or IDREFS attribute whose instances in D refer to instances of 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*.

□

We can now define the set of immediate neighbours of an x-component x_S.

Definition 9. Let D be an XML document and let x_S be an x-component of D. We define the immediate neighborhood of x_S, denoted by $\mathcal{N}^I_{x_S}$, as:

$$\mathcal{N}^I_{x_S} = \{x_T |\ x_T \in XCompSet(D), near(x_S, x_T) = true\}$$

An x-component in $\mathcal{N}^I_{x_S}$ is called *immediate neighbour* of x_S; we say also that x_T is *directly reachable* from x_S. □

Generally, for each x-component, we are interested to all neighborhood levels and not only to the immediate one. In order to specify these levels, it is necessary to introduce some definitions.

Definition 10. Let D be an XML document; let x_S be an x-component of D and let x_T be an element of $\mathcal{N}^I_{x_S}$; the *Immediate Neighborhood Degree* from x_S to x_T, denoted by $IND(x_S, x_T)$, is defined as follows[3]:

[3] Observe that, since $x_T \in \mathcal{N}^I_{x_S}$, either $veryclose(x_S, x_T) = true$ or $close(x_S, x_T) = true$.

$$IND(x_S, x_T) = \begin{cases} 0 \text{ if } x_S = x_T \text{ or } veryclose(x_S, x_T) = true \\ 1 \text{ if } close(x_S, x_T) = true \end{cases}$$ □

The following definitions extend all concepts introduced above to x-components not directly reachable each other.

Definition 11. Let D be an XML document and let x_S and x_T be two x-components of D such that $x_T \notin \mathcal{N}^I_{x_S}$. We say that x_T is *indirectly reachable* from x_S if there exists a sequence of *distinct* x-components $\sigma = \{x_1, x_2, \ldots, x_n\}$ such that $x_S = x_1, near(x_1, x_2) = near(x_2, x_3) = \ldots = near(x_{n-1}, x_n) = true, x_n = x_T$. □

Definition 12. Let D be an XML document and let x_S and x_T be two x-components of D such that x_T is indirectly reachable from x_S by means of the sequence of distinct x-components $\sigma = \{x_1, x_2, \ldots, x_n\}$. The cost of an indirect connection from x_S to x_T by the sequence σ (denoted by $\mathcal{C}^{\sigma}_{ST}$) is defined as $\mathcal{C}^{\sigma}_{ST} = \sum_{i=1}^{n-1} IND(x_i, x_{i+1})$ □

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

Definition 13. Let D be an XML document and let x_S and x_T be two x-components of D. The connection cost from x_S to x_T, denoted by $CC(x_S, x_T)$, is defined as:

$$CC(x_S, x_T) = \begin{cases} IND(x_S, x_T) & \text{if } x_T \text{ is directly reachable from } x_S \\ min_{\sigma}\ \mathcal{C}^{\sigma}_{ST} & \text{if } x_T \text{ is indirectly reachable from } x_S \\ \infty & \text{if } x_T \text{ is not reachable from } x_S \end{cases}$$

Here $min_{\sigma}\ \mathcal{C}^{\sigma}_{ST}$ is computed over all possible sequences of distinct x-components connecting x_S and x_T. □

The following proposition is particularly important and its proof is immediate.

Proposition 14. Let D be an XML document and let x_S and x_T be two x-components of D. If $CC(x_S, x_T) \neq \infty$, then $0 \leq CC(x_S, x_T) \leq |XCompSet(D)| - 1$ □

Now, we are provided with all tools necessary to define the concept of neighborhood of an x-component.

Definition 15. Let D be an XML document and let x_S be an x-component of D. The j^{th} neighborhood of x_S is defined as:

$$neighborhood(x_S, j) = \{x_T |\ x_T \in XCompSet(D), CC(x_S, x_T) = j\}$$ □

It is worth pointing out that if $x_T \in neighborhood(x_S, j)$ then $x_T \notin neighborhood(x_S, l)$ for $l \neq j$.

Observe that the functions $veryclose(x_S, s_T)$ and $close(x_S, x_T)$, and consequently the definition of neighborhood, which is based on them, are not symmetric. In other words, given an x-component, we put its ancestors but not its descendants in its neighborhood. Such a choice is motivated by considering that, given an x-component x_C, its descendants represent its properties whereas its ancestors denote concepts which x_C is a property of. As a further example clarifying this choice consider the following situation: an XML source S_1 has an element *Professor*, having an attribute *Birthdate*; an XML source S_2 has an element *Student*, having an attribute *Birthdate*. Clearly these attributes are synonymous even if their ancestors are different.

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

Definition 16. Let D be an XML document and let $\mathcal{T}$ be the corresponding DTD. The *X-Dist-Graph* relative to D and $\mathcal{T}$ is an oriented labeled graph defined as $XG(D, \mathcal{T}) = \langle N(\mathcal{T}), A(D, \mathcal{T})\rangle$. Here, $N(\mathcal{T})$ is the set of nodes of $XG(D, \mathcal{T})$; there is a node in $XG(D, \mathcal{T})$ for each x-component of $\mathcal{T}$. $A(D, \mathcal{T})$ is the set of arcs of $XG(D, \mathcal{T})$; there is an arc $\langle N_S, N_T, f_{ST}\rangle$ in $XG(D, \mathcal{T})$ 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 $XG(D, \mathcal{T})$ corresponding to x_S (resp., x_T) and $f_{ST} = IND(x_S, x_T)$. $\square$

The following proposition specifies the computational complexity of the construction of $XG(D, \mathcal{T})$; its proof can be found in the Appendix.

Proposition 17. Let D be an XML document and let $\mathcal{T}$ be the corresponding DTD. Let n be the number of x-components of D and let N_{inst} be the number of instances of D. The worst case time complexity for constructing $XG(D, \mathcal{T})$ from D and $\mathcal{T}$ 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 instances which the instance of an IDREFS attribute refers to, it is necessary to examine the document, since the DTD does not provide such an information. As a consequence, the dependency of the computational complexity on N_{inst} cannot be avoided. However, we point out that the quadratic dependency on N_{inst} is mainly a theoretical result; indeed, it derives from the consideration that each IDREFS attribute instance might refer to N_{inst} element instances. Actually, in real situations, each IDREFS attribute refers to a very limited number of instances; as a consequence, the dependency of the computational complexity on 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 document D; its proof is presented in the Appendix.

Theorem 18. Let $XG(D, \mathcal{T})$ be the XS-Graph associated with an XML document D and the corresponding DTD $\mathcal{T}$ and let n be the number of x-components of D. The worst case time complexity for computing the set $\{neighborhood(x_i, j) | x_i \in XCompSet(D), 0 \leq j \leq n\}$ is $O(n^3)$.[4] $\square$

From Definition 15 it is straightforward to introduce the concept of extended neighborhood that plays a fundamental role in the interscheme property derivation.

Definition 19. Let D be an XML document, let x_S be an x-component of D and let j be an integer greater than or equal to 0. The j^{th} *extended neighborhood* of x_S is defined as:

$$ext\text{-}neighborhood(x_S, j) = \bigcup_{k=0}^{j} neighborhood(x_S, k) \qquad \square$$

The following example helps to illustrate the concepts presented in this section.

Example 2. Consider the XML document U_1, shown in Figure 1, and the corresponding DTD, shown in Figure 2. Here, $veryclose(Professor, Name) = true$ because $Name$ is an attribute of $Professor$; analogously $close(Professor, e\text{-}mail) = true$ because *e-mail* is a sub-element of *Professor* and $close(Professor, Course) = true$ because there is an instance of the IDREFS attribute *Teaches_in* of *Professor* referring to instances of *Course*. Finally, $near(Student, Test) = true$ because $close(Student, Test) = true$.

If we consider the x-component *Professor* we have:

$$\begin{aligned}\mathcal{N}^I_{Professor} = \{&Identifier_{\langle Professor\rangle}, Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle},\\ &Salary_{\langle Professor\rangle}, Belongs\_to_{\langle Professor\rangle}, Teaches\_in_{\langle Professor\rangle},\\ &e\text{-}mail, Phone, Course, Department\}\end{aligned}$$

Here, $IND(Professor, Name_{\langle Professor\rangle}) = 0$ because $veryclose(Professor, Name_{\langle Professor\rangle}) = true$ whereas $IND(Professor, Course) = 1$ since $close(Professor, Course) = true$. Finally, consider the sequence of x-components $\sigma = \{Professor, Course, Student, Test\}$; in this case $\mathcal{C}^{\sigma}_{Professor,Test} = 3$ because $IND(Professor, Course) = IND(Course, Student) = IND(Student, Test) = 1$ and, consequently, $CC(Professor, Test) = 3$. Some of the neighborhoods of $Professor$ are the following:

$$\begin{aligned}neighborhood(Professor, 0) = \{&Professor, Identifier_{\langle Professor\rangle},\\ &Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle}, Salary_{\langle Professor\rangle},\\ &Belongs\_to_{\langle Professor\rangle}, Teaches\_in_{\langle Professor\rangle}\}\\ neighborhood(Professor, 1) = \{&e\text{-}mail, Phone, Department,\\ &Identifier_{\langle Department\rangle}, Director_{\langle Department\rangle}, Floor_{\langle Department\rangle},\\ &Course, Identifier_{\langle Course\rangle}, Name_{\langle Course\rangle},\\ &Student\_Number_{\langle Course\rangle}, Argument_{\langle Course\rangle}, Duration_{\langle Course\rangle},\\ &Attended\_by_{\langle Course\rangle}, Teached\_by_{\langle Course\rangle}\}\end{aligned}$$

[4] Observe that, as a consequence of Proposition 14, the maximum degree of a neighborhood of an x-component of an XML document D is equal to $|XCompSet(D)|$.

To illustrate, $Director_{\langle Department \rangle}$ belongs to $neighborhood(Professor, 1)$ because, if we consider the sequence $\sigma = \{Professor, Department, Director\}$, we have that $\mathcal{C}^{\sigma}_{Professor,Director} = IND(Professor, Department) + IND(Department, Director) = 1 + 0 = 1$ and $\mathcal{C}^{\sigma}_{Professor,Director}$ is the minimum connection cost between *Professor* and *Director*. The other neighborhoods can be determined analogously.

As an example of *ext-neighborhood*, if we consider the x-component $Professor$, we have that:

$$
\begin{aligned}
&ext\text{-}neighborhood(Professor, 1) = \{Professor, Identifier_{\langle Professor \rangle},\\
&\quad Name_{\langle Professor \rangle}, Birthdate_{\langle Professor \rangle}, Salary_{\langle Professor \rangle},\\
&\quad Belongs\_to_{\langle Professor \rangle}, Teaches\_in_{\langle Professor \rangle}, e\text{-}mail, Phone, Department,\\
&Identifier_{\langle Department \rangle}, Director_{\langle Department \rangle}, Floor_{\langle Department \rangle}, Course,\\
&\quad Identifier_{\langle Course \rangle}, Name_{\langle Course \rangle}, Student\_Number_{\langle Course \rangle},\\
&\quad Argument_{\langle Course \rangle}, Duration_{\langle Course \rangle}, Attended\_by_{\langle Course \rangle},\\
&\quad Teached\_by_{\langle Course \rangle}\}
\end{aligned}
$$

□

4 Extraction of interscheme properties among x-components

In this section we illustrate how *X-Global* computes interscheme properties among x-components belonging to different XML documents. As pointed out in the Introduction, the knowledge of these properties is crucial for the integration activity. In this paper we shall consider the following interscheme properties, possibly holding between two x-components x_A and x_B, belonging to different XML documents:

- *synonymy*: it indicates that x_A and x_B represent the same concept and have either the same or compatible types;
- *homonymy*: it denotes that x_A and x_B represent different concepts yet having both the same name and either the same or compatible types;
- *type conflict*: it indicates that x_A and x_B represent the same concept yet having incompatible types.

As pointed out in the Introduction, our technique for computing interscheme properties is semantic [3, 11, 28] in that, in order to determine the meaning of an x-component, it examines the "context" which it has been defined in; in our approach the context of an x-component consists of the set of x-components belonging to its neighborhoods.

Our approach for computing interscheme properties requires the presence of a thesaurus storing lexical synonymies existing among the terms of a language. In particular, we have exploited the English language and WordNet [26][5].

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

The extraction of interscheme properties is carried out in two phases, namely: *(i)* the derivation of similarities possibly existing among x-components belonging to different XML documents; *(ii)* the extraction of synonymies, homonymies and type conflicts starting from both the set of derived similarities and the x-component types. In the next two sub-sections we shall describe each of these steps into detail.

4.1 Derivation of similarities between x-components

As previously pointed out, in order to verify if two x-components x_{1_j}, belonging to an XML document D_1, and x_{2_k}, belonging to an XML document D_2, are similar, it is necessary to verify if their neighborhoods are similar.

In order to determine if two neighborhoods, say $ext\text{-}neighborhood(x_{1_j}, u)$ and $ext\text{-}neighborhood(x_{2_k}, u)$, are similar, 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 $ext\text{-}neighborhood(x_{1_j}, u)$ and $ext\text{-}neighborhood(x_{2_k}, u)$.

More specifically, let $BG(x_{1_j}, x_{2_k}, u) = \langle N, A \rangle$ be the bipartite graph associated with
$ext\text{-}neighborhood(x_{1_j}, u)$ and $ext\text{-}neighborhood(x_{2_k}, u)$ (in the following we shall use the notation BG instead of $BG(x_{1_j}, x_{2_k}, u)$ when this is not confusing). In BG, $N = P \cup Q$ represents the set of nodes; there is a node in P (resp., Q) for each x-component in $ext\text{-}neighborhood(x_{1_j}, u)$ (resp., $ext\text{-}neighborhood(x_{2_k}, u)$). A is the set of arcs; there is an arc between $p_e \in P$ and $q_f \in Q$ if there exists a lexical similarity between the names of the x-components associated with p_e and q_f. The maximum weight matching for BG is a set $A' \subseteq A$ of edges such that, for each node $x \in P \cup Q$, there is at most one edge of A' incident onto x and $|A'|$ is maximum (for algorithms solving the maximum weight matching problem, see [12]).

The objective function associated with the maximum weight matching is computed as:

$$\phi_{BG} = \begin{cases} 1 & \textit{if } \frac{2|A'|}{|P|+|Q|} > th_\phi \\ 0 & \textit{otherwise} \end{cases}$$

Here th_ϕ is a suitable threshold. In the prototype implementing our approach it has been set to 0.5; the various experiments we have performed for testing our approach, whose main results are described in Section 6, confirm the appropriateness of this value.

The definition of ϕ_{BG} allows us to specify when two extended neighborhoods are similar. In particular, we say that $ext\text{-}neighborhood(x_{1_j}, u)$ and $ext\text{-}neighborhood(x_{2_k}, u)$ are similar if $\phi_{BG(x_{1_j}, x_{2_k}, u)} = 1$.

Vice versa, if $\phi_{BG(x_{1_j},x_{2_k},u)} = 0$, we say that *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u) are dissimilar.

The following definition exploits extended neighborhood similarity introduced above for determining when two x-components are similar. It allows a flexibility level to be associated with each derived similarity.

Definition 20. Let D_1 and D_2 be two XML documents and let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be a non-negative integer. We say that x_{1_j} and x_{2_k} are *similar* with a flexibility level equal to u, indicated by $similar(x_{1_j}, x_{2_k}, u)$, if *ext-neighborhood*$(x_{1_j}, u)$ and *ext-neighborhood*(x_{2_k}, u) are similar. We assume that if x_{1_j} and x_{2_k} are similar with a flexibility level equal to u, then they are also similar with every flexibility level v such that $u \leq v \leq n$, where n is the maximum between $|XCompSet(D_1)|$ and $|XCompSet(D_2)|$. $\square$

If all neighborhoods of x_{1_j} and x_{2_k} have been examined and no similarity has been found, we say that x_{1_j} and x_{2_k} are similar with flexibility level ∞, i.e., they are dissimilar. As far as this last case is concerned, the following proposition, whose proof is shown in the Appendix, is extremely important.

Proposition 21. Let D_1 and D_2 be two XML documents; let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2); finally, let n be the maximum between $|XCompSet(D_1)|$ and $|XCompSet(D_2)|$. If x_{1_j} and x_{2_k} are not similar with a flexibility level equal to n, then they are dissimilar. $\square$

A function *similar* 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 similar with a flexibility level equal to u, $false$ otherwise.

Definition 20 requires to specify when two neighborhoods, We present now the following theorems, allowing the computational complexity associated with the extraction of x-components' similarities to be determined; their proofs are presented in the Appendix.

Theorem 22. Let D_1 and D_2 be two XML documents. Let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be an integer greater than or equal to 0. Finally, let p be the maximum between the cardinalities of *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u). The computational cost for evaluating if *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u) are similar is $O(p^3)$. $\square$

Theorem 23. Let D_1 and D_2 be two XML documents. Let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be the selected flexibility level. Finally, let p be the maximum between the cardinalities of *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u). The worst case time complexity for computing $similar(x_{1_j}, x_{2_k}, u)$ is $O((u + 1) \times p^3)$. $\square$

It is worth pointing out that the *semantic* concept similarity described above is heuristic. This is commonly accepted in the literature because no formal framework can be defined for identifying this kind of properties. This is motivated by considering that the semantic similarity detection involves the way the content of information sources into consideration is represented by designers and perceived by users. As a further motivation consider that, in the literature, it is generally recognized that purely structural considerations (i.e., the only ones that can be not subjective) do not suffice to determine semantic similarities [3, 11]. As a consequence, it is not possible to define a completely formal model for handling the detection of semantic concept similarities.

4.2 Derivation of interscheme properties

In the previous section we have shown how x-component similarities can be derived from a pair of XML sources. However, their knowledge is not sufficient on its own for determining interscheme properties. As a matter of fact, the knowledge about the types of x-components into consideration is as important as the knowledge of their similarity degree. Indeed, two x-components might be semantically similar and have the same type: in this case a synonymy holds between them. Vice versa, a semantic similarity could exists between two x-components having different types: in this case a type conflict exists between them. In order to formally specify these concepts, the following definitions are needed.

Definition 24. Let D_1 and D_2 be two XML documents and let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be a non-negative integer. We say that a *synonymy* holds between x_{1_j} and x_{2_k} with a flexibility level equal to u if $similar(x_{1_j}, x_{2_k}, u) = true$ and no type incompatibility exists between them. □

A boolean function *synonymy* can be defined, which receives two x-components x_{1_j} and x_{2_k} and an integer u and returns $true$ if there exists a synonymy between x_{1_j} and x_{2_k} with a flexibility level equal to u; *synonymy* returns $false$ otherwise.

Definition 25. Let D_1 and D_2 be two XML documents and let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be a non-negative integer. We say that an homonymy holds between x_{1_j} and x_{2_k} with a flexibility level equal to u if: *(i)* $similar(x_{1_j}, x_{2_k}, u) = false$; *(ii)* x_{1_j} and x_{2_k} have the same name; *(iii)* no type incompatibility exists between them. □

A boolean function *homonymy* can be defined, 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 flexibility level equal to u; *homonymy* returns $false$ otherwise.

Definition 26. Let D_1 and D_2 be two XML documents and let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be a non-negative integer. We say that there exists a *first kind type conflict* between x_{1_j} and x_{2_k} with a flexibility level equal to u if $similar(x_{1_j}, x_{2_k}, u) = true$ and a first kind type incompatibility exists between x_{1_j} and x_{2_k}. □

Definition 27. Let D_1 and D_2 be two XML documents and let x_{1_j} (resp., x_{2_k}) be one of the attributes of an element x'_{1_j} (resp., x'_{2_k}) in D_1 (resp., D_2). Let u be a non-negative integer. We say that there exists a *second kind type conflict* between x_{1_j} and x_{2_k} with a flexibility level equal to u if: *(i)* $similar(x'_{1_j}, x'_{2_k}, u) = true$, *(ii)* x_{1_j} and x_{2_k} have the same name and *(iii)* a second kind type incompatibility exists between x_{1_j} and x_{2_k}. □

A boolean function fk-*tconflict* (resp., *sk-tconflict*) can be defined, which receives two x-components (resp., two attributes) x_{1_j} and x_{2_k} and an integer u and returns $true$ if there exists a first kind (resp., a second kind) type conflict between x_{1_j} and x_{2_k} with a flexibility level equal to u; fk-*tconflict* (resp., *sk-tconflict*) returns $false$ otherwise.

Theorem 28. Let D_1 and D_2 be two XML documents. Let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be a selected flexibility level. Finally, let p be the maximum between the cardinalities of *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u). The worst case time complexity for determining if there exists a synonymy (resp., an homonymy, a first kind type conflict, a second kind type conflict) between x_{1_j} and x_{2_k} with a flexibility level equal to u is $O((u + 1) \times p^3)$.
Proof

Immediate from Theorem 23 and Definition 24 (resp., Definition 25, Definition 26, Definition 27). □

Corollary 29. Let D_1 and D_2 be two XML documents. Let u be a selected flexibility level. Let m be the maximum between $|XCompSet(D_1)|$ and $|XCompSet(D_2)|$. Finally, let q be the maximum cardinality relative to a neighborhood of D_1 or D_2. The worst case time complexity for deriving all interscheme properties existing, at the flexibility level u, between D_1 and D_2 is $O((u + 1) \times q^3 \times m^2)$. □

Example 3. Consider the XML documents U_1 and U_2, shown in Figures 1 and 3, and the corresponding DTDs, presented in Figures 2 and 4.

Consider now the x-components $Professor_{[U_1]}$[6] and $Professor_{[U_2]}$. In order to check if they are similar with flexibility level 0, it is necessary to compute $similar(Professor_{[U_1]}, Professor_{[U_2]}, 0)$. Now,

[6] Here and in the following we use the notation $x_{[D]}$ to indicate the x-component x of the XML document D.

```
<?xml version=‘‘1.0’’?>
<!DOCTYPE University SYSTEM "University.dtd">
<University>
      <Professor Id=‘‘P1’’ Name=‘‘Smith’’ Birthdate=‘‘July 12 1950’’ Teaches_In=‘‘C1 C2’’ Papers=‘‘Pa1 Pa2’’>
           <Phone> +39 06 000001 </Phone>
           <e-mail> smith@hotmail.com </e-mail>
           <e-mail> smith@uniroma1.it </e-mail>
           <Project Starting_Date=‘‘April 3 1999’’ Ending_Date=‘‘April 3 2001’’> Project1 </Project>
      </Professor>
      <Professor Id=‘‘P2’’ Name=‘‘Brown’’ Birthdate=‘‘May 6 1958’’ Birthplace=‘‘Milan’’ Teaches_In=‘‘C3’’
               Papers=‘‘Pa1’’>
           <Phone> +39 02 00011 </Phone>
           <Phone> +39 06 33333 </Phone>
           <e-mail> brown@hotmail.com </e-mail>
           <Project Ending_Date=‘‘September 5 2002’’> Project2 </Project>
           <Project Starting_Date=‘‘November 3 1998’’ Ending_Date=‘‘February 4 2002’’> Project3 </Project>
           <Project> Project4 </Project>
      </Professor>
      <Paper Id=‘‘Pa1’’ Authors=‘‘P1 P2’’ Volume=‘‘VolA’’ Pages=‘‘0-4’’> Paper1 </Paper>
      <Paper Id=‘‘Pa2’’ Authors=‘‘P1’’ Type=‘‘Conference’’ Volume=‘‘VolB’’ Pages=‘‘14-20’’> Paper2 </Paper>
      <Course Id=‘‘C1’’ Name=‘‘Foundamentals of Computer Science’’ Responsible=‘‘P1’’ >
           <Year> 1 </Year>
           <Student_Number> 300 </Student_Number>
           <Argument> Computer Architecture </Argument>
           <Argument> Java </Argument>
      </Course>
      <Course Id=‘‘C2’’ Name=‘‘Databases’’ Responsible=‘‘P1’’ Propaedeutic_Courses=‘‘C1’’>
           <Year> 4 </Year>
           <Argument> Relational Databases </Argument>
           <Argument> SQL </Argument>
           <Argument> Object Oriented Databases </Argument>
      </Course>
      <Course Id=‘‘C3’’ Name=‘‘Information Systems’’ Responsible=‘‘P2’’ Propaedeutic_Courses=‘‘C1 C2’’>
           <Argument> Cooperative Information Systems </Argument>
           <Argument> Data Warehouses </Argument>
      </Course>
      <Student Id=‘‘S1’’ Name=‘‘Rossi’’ Birthdate=‘‘May 5 1980’’ Birthplace=‘‘Rome’’ Attended_Courses=‘‘C1’’>
           <Address> Rome </Address>
      </Student>
      <Student Id=‘‘S2’’ Name=‘‘Verdi’’ Birthdate=‘‘April 20 1976’’ Birthplace=‘‘Neaples’’
               Enrollment_Year=‘‘1995’’ Exams=‘‘E1’’ Attended_Courses=‘‘C2’’>
           <Address> Rome </Address>
      </Student>
      <Student Id=‘‘S3’’ Name=‘‘Bianchi’’ Birthdate=‘‘February 18 1975’’ Enrollment_Year=‘‘1994’’
               Exams=‘‘E2 E3’’ Attended_Courses=‘‘C3’’>
           <Address> Rome </Address>
           <Thesis>
                <Title> Title1 </Title>
                <Topic> Topic1 </Topic>
           </Thesis>
      </Student>
      <Student Id=‘‘S4’’ Name=‘‘Johnson’’ Birthdate=‘‘December 20 1976’’ Birthplace=‘‘Neaples’’
               Enrollment_Year=‘‘1995’’ Exams=‘‘E4’’ Attended_Courses=‘‘C2 C3’’>
           <Address> Rome </Address>
      </Student>
      <Exam Id=‘‘E1’’ Student=‘‘S2’’ Course=‘‘C1’’>
           <Date> April 3 1996 </Date>
           <Grade> 23 </Grade>
      </Exam>
      <Exam Id=‘‘E2’’ Student=‘‘S3’’ Course=‘‘C1’’>
           <Date> May 6 1995 </Date>
           <Grade> 27 </Grade>
      </Exam>
      <Exam Id=‘‘E3’’ Student=‘‘S3’’ Course=‘‘C2’’>
           <Date> May 10 1999 </Date>
           <Grade> 29 </Grade>
      </Exam>
      <Exam Id=‘‘E4’’ Student=‘‘S4’’ Course=‘‘C1’’>
           <Date> September 5 1996 </Date>
           <Grade> 30 </Grade>
      </Exam>
</University>
```

Figure 3: The XML document U_2 representing a University

```
<!ELEMENT University (Professor+, Course+, Student+)>
<!ELEMENT Professor (Phone+, e-mail*, Project*)>
   <!ATTLIST Professor
      Id ID #REQUIRED
      Name CDATA #REQUIRED
      Birthdate CDATA #IMPLIED
      Birthplace CDATA #IMPLIED
      Teaches_in IDREFS #IMPLIED
      Papers IDREFS #IMPLIED
   >
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
<!ELEMENT Project (#PCDATA)>
   <!ATTLIST Project
      Starting_Date CDATA #IMPLIED
      Ending_Date CDATA #IMPLIED
   >
<!ELEMENT Paper (#PCDATA)>
   <!ATTLIST Paper
      Id ID #REQUIRED
      Authors IDREFS #REQUIRED
      Type (Journal|Conference) ''Journal''
      Volume CDATA #REQUIRED
      Pages CDATA #REQUIRED
   >
<!ELEMENT Course (Year?, Student_Number?, Argument+)>
   <!ATTLIST Course
      Id ID #REQUIRED
      Name CDATA #REQUIRED
      Responsible IDREF #REQUIRED
      Propaedeutic_Courses IDREFS #IMPLIED
   >
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Student_Number (#PCDATA)>
<!ELEMENT Argument (#PCDATA)>
<!ELEMENT Student (Address, Thesis?)>
   <!ATTLIST Student
      Id ID #REQUIRED
      Name CDATA #REQUIRED
      Birthdate CDATA #IMPLIED
      Birthplace CDATA #IMPLIED
      Enrollment_Year CDATA #IMPLIED
      Exams IDREFS #IMPLIED
      Attended_Courses IDREFS #IMPLIED
   >
<!ELEMENT Address (#PCDATA)>
<!ELEMENT Thesis (Title, Topic+)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Topic (#PCDATA)>
<!ELEMENT Exam (Date, Grade)>
   <!ATTLIST Exam
      Id ID #REQUIRED
      Student IDREF #REQUIRED
      Course IDREF #REQUIRED
   >
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Grade (#PCDATA)>
```

Figure 4: The DTD of the XML document U_2

$$\begin{aligned}&\textit{ext-neighborhood}(Professor_{[U_1]},0) = \{Professor, Identifier_{\langle Professor\rangle},\\&\quad Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle}, Salary_{\langle Professor\rangle},\\&\quad Belongs\_to_{\langle Professor\rangle}, Teaches\_in_{\langle Professor\rangle}\}\\&\textit{ext-neighborhood}(Professor_{[U_2]},0) = \{Professor, Id_{\langle Professor\rangle},\\&\quad Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle}, Birthplace_{\langle Professor\rangle},\\&\quad Teaches\_in_{\langle Professor\rangle}, Papers_{\langle Professor\rangle}\}\end{aligned}$$

The maximum weight matching computed by the function *similar* is illustrated in Figure 5. In this case, $\frac{2|A'|}{|P|+|Q|} = \frac{2\times 5}{7+7} = 0.71 > th_\phi$[7]; therefore, ϕ_{BG} returns 1 and $similar(Professor_{[U_1]}, Professor_{[U_2]}, 0) = true$. As a consequence, $Professor_{[U_1]}$ and $Professor_{[U_2]}$ are similar with flexibility level equal to 0.

As a further example, consider the x-components $Course_{[U_1]}$ and $Course_{[U_2]}$. In this case:

$$\begin{aligned}&\textit{ext-neighborhood}(Course_{[U_1]},0) = \{Course, Identifier_{\langle Course\rangle}, Name_{\langle Course\rangle},\\&\quad Student\_Number_{\langle Course\rangle}, Argument_{\langle Course\rangle}, Duration_{\langle Course\rangle},\\&\quad Attended\_by_{\langle Course\rangle}, Teached\_by_{\langle Course\rangle}\}\\&\textit{ext-neighborhood}(Course_{[U_2]},0) = \{Course, Id_{\langle Course\rangle}, Name_{\langle Course\rangle},\\&\quad Responsible_{\langle Course\rangle}, Propaedeutic\_Courses_{\langle Course\rangle}\}\end{aligned}$$

The computation of the maximum weight matching returns $\frac{2|A'|}{|P|+|Q|} = \frac{2\times 3}{8+5} = 0.46 < th_\phi$; thus, $\phi_{BG} = 0$ and $Course_{[U_1]}$ and $Course_{[U_2]}$ are not similar with flexibility level 0. Now we check if they are similar with flexibility level 1.

$$\begin{aligned}&\textit{ext-neighborhood}(Course_{[U_1]},1) = \{Course, Identifier_{\langle Course\rangle}, Name_{\langle Course\rangle},\\&\quad Student\_Number_{\langle Course\rangle}, Argument_{\langle Course\rangle}, Duration_{\langle Course\rangle},\end{aligned}$$

[7] Remember that we have set $th_\phi = 0.5$ (see Section 4.1).

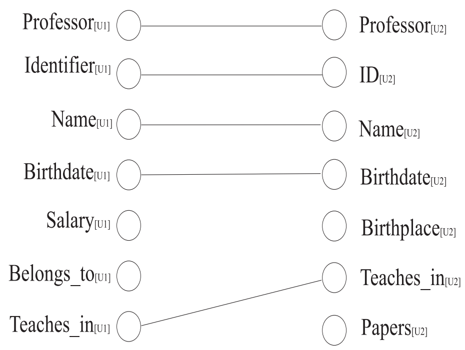

Figure 5: The maximum weight matching computed by $similar(Professor_{[U_1]}, Professor_{[U_2]}, 0)$

$$\begin{array}{l} \quad Attended\_by_{\langle Course\rangle}, Teached\_by_{\langle Course\rangle}, Student, Identifier_{\langle Student\rangle}, \\ \quad Name_{\langle Student\rangle}, Average\_Mark_{\langle Student\rangle}, Attends_{\langle Student\rangle}, Professor, \\ \quad Birthdate_{\langle Student\rangle}, Enrollment\_Year_{\langle Student\rangle}, Tutor_{\langle Student\rangle}, \\ \quad Identifier_{\langle Professor\rangle}, Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle}, \\ \quad Salary_{\langle Professor\rangle}, Belongs\_to_{\langle Professor\rangle}, Teaches\_in_{\langle Professor\rangle}\} \\ ext\text{-}neighborhood(Course_{[U_2]}, 1) = \{Course, Id_{\langle Course\rangle}, Name_{\langle Course\rangle}, \\ \quad Responsible_{\langle Course\rangle}, Propaedeutic\_Courses_{\langle Course\rangle}, Professor, \\ \quad Id_{\langle Professor\rangle}, Name_{\langle Professor\rangle}, Birthdate_{\langle Professor\rangle}, \\ \quad Birthplace_{\langle Professor\rangle}, Teaches\_in_{\langle Professor\rangle}, Papers_{\langle Professor\rangle}, \\ \quad Year, Student\_Number, Argument\} \end{array}$$

In this case, ϕ_{BG} returns 1 because $\frac{2|A'|}{|P|+|Q|} = \frac{2\times 10}{23+15} = 0.53 > th_\phi$. This allows us to conclude that $Course_{[U_1]}$ and $Course_{[U_2]}$ are similar with a flexibility level equal to 1.

Finally, $synonymy(Professor_{[U_1]}, Professor_{[U_2]}, 0)$ returns $true$ because $similar(Professor_{[U_1]}, Professor_{[U_2]}, 0) = true$ and both $Professor_{[U_1]}$ and $Professor_{[U_2]}$ are elements. Vice versa, even if the x-components $Argument_{[U_1]}$ and $Argument_{[U_2]}$ are similar with a flexibility level equal to 0, they are not synonymous; indeed, a first kind type incompatibility holds between them since $Argument_{[U_1]}$ is an attribute and $Argument_{[U_2]}$ is an element. Finally, *homonymy*$(Course_{[U_1]}, Course_{[U_2]}, 0) = true$ since $similar(Course_{[U_1]}, Course_{[U_2]}, 0) = false$, no type incompatibility exists because $Course_{[U_1]}$ and $Course_{[U_2]}$ and they have the same name. On the contrary, $homonymy(Course_{[U_1]}, Course_{[U_2]}, 1) = false$.

□

5 The Integration Task

In this section we illustrate the integration algorithm underlying *X-Global*. It basically consists of two phases: the first one receives two XML documents D_1

and D_2, their DTDs $\mathcal{T}_1$ and $\mathcal{T}_2$ and a flexibility level u and returns a global DTD $\mathcal{T}_\mathcal{G}$ obtained by merging all the concepts of $\mathcal{T}_1$ and $\mathcal{T}_2$. The second phase generates a global XML document D_G which conforms to $\mathcal{T}_\mathcal{G}$ and merges all information stored in D_1 and D_2.

The algorithm first computes synonymies, homonymies and type conflicts existing among the x-components of D_1 and D_2 with a flexibility level equal to u; these properties are stored in the Synonymy Dictionary SD, the Homonymy Dictionary HD, the First Kind Type Conflict Dictionary $FKTCD$ and the Second Kind Type Conflict Dictionary $SKTCD$. Each dictionary is a collection of pairs of the form $[x_{1_j}, x_{2_k}]$, where x_{1_j} and x_{2_k} are the involved x-components.

After all interscheme properties have been derived, the algorithm normalizes D_1 and D_2 for solving all first kind type conflicts. In order to understand how this normalization activity is carried out, assume that a first kind type conflict between the element x_{1_j} of D_1 and the attribute x_{2_k} of D_2 holds; moreover, assume that x_{2_k} is one of the attributes of an element x_{2_E} in D_2. Our approach carries out the following operations for solving the type conflict between x_{1_j} and x_{2_k}:

- It creates an element x'_{2_k} in D_2, having the same name as x_{2_k}, whose type definition is as follows:

```
<!ELEMENT x'_{2_k} (#PCDATA)>
<!ATTLIST x'_{2_k}
   Identifier ID #REQUIRED
>
```

  In this way, each value of x_{2_k} can be "mapped" into the #PCDATA component of x'_{2_k}; values of the attribute *Identifier* are automatically assigned during the normalization.

- It creates an IDREF attribute x_{2_A} of x_{2_E} in such a way that each instance of x_{2_E} refers, via the attribute x_{2_A}, to an instance of x'_{2_k}.

- It removes x_{2_k} from D_2.

- It eliminates the pair $[x_{1_j}, x_{2_k}]$ from $FKTCD$.

- It adds the pair $[x_{1_j}, x'_{2_k}]$ to SD.

In the same way all the other first kind type conflicts are solved and transformed into synonymies.

After D_1 and D_2 have been normalized, the construction of $\mathcal{T}_\mathcal{G}$ can start. The first step of this activity consists in the creation of an x-component x_{G_j} (resp., x_{G_k}) in $\mathcal{T}_\mathcal{G}$ for each component x_{1_j} (resp., x_{2_k}) of $\mathcal{T}_1$ (resp., $\mathcal{T}_2$). x_{G_j} (resp., x_{G_k}) is a duplicate of x_{1_j} (resp., x_{2_k}) in $\mathcal{T}_\mathcal{G}$; in particular, it has the same name, the same type, the same attributes (if any) participating with the same cardinalities, the same sub-elements (if any) participating with the same cardinalities[8] in $\mathcal{T}_1$ (resp., $\mathcal{T}_2$). In the following x_{G_j} (resp., x_{G_k}) will be called the *global duplicate* of x_{1_j} (resp., x_{2_k}) in $\mathcal{T}_\mathcal{G}$ whereas x_{1_j} (resp., x_{2_k}) will be referred as the *local copy* of x_{G_j} (resp., x_{G_k}) in $\mathcal{T}_1$ (resp., $\mathcal{T}_2$).

At the end of this step $\mathcal{T}_\mathcal{G}$ could contain some redundancies and/or ambiguities. In order to remove them and, consequently, to refine $\mathcal{T}_\mathcal{G}$, it is necessary to examine SD, HD and $SKTCD$ and to perform some tasks for each of the properties they store.

First element synonymies registered in SD must be considered. In particular, for each synonymy $[x_{1_j}, x_{2_k}]$ existing in SD such that both x_{1_j} and x_{2_k} are elements, the corresponding global duplicates $x_{G_{1_j}}$ and $x_{G_{2_k}}$ must be merged into an element $x_{G_{jk}}$. In order to perform this task, the following steps must be executed:

1. The name of $x_{G_{jk}}$ is set to either the name of $x_{G_{1_j}}$ or that of $x_{G_{2_k}}$.

2. If the content specification of both $x_{G_{1_j}}$ and $x_{G_{2_k}}$ is ANY (resp., EMPTY, #PCDATA) then the content specification of $x_{G_{jk}}$ is ANY (resp., EMPTY, #PCDATA).

3. If the content specification of $x_{G_{1_j}}$ (resp., $x_{G_{2_k}}$) is ANY and that of $x_{G_{2_k}}$ (resp., $x_{G_{1_j}}$) is different from ANY then the content specification of $x_{G_{jk}}$ is the OR of the content specifications of $x_{G_{1_j}}$ and $x_{G_{2_k}}$ (see Definition 4).

4. If the content specification of $x_{G_{1_j}}$ (resp., $x_{G_{2_k}}$) is EMPTY and that of $x_{G_{2_k}}$ (resp., $x_{G_{1_j}}$) is different from ANY or EMPTY then the content specification of $x_{G_{jk}}$ is the OR of the content specifications of $x_{G_{1_j}}$ and $x_{G_{2_k}}$.

5. If the content specification of $x_{G_{1_j}}$ (resp., $x_{G_{2_k}}$) is #PCDATA and that of $x_{G_{2_k}}$ (resp., $x_{G_{1_j}}$) is different from ANY, EMPTY and #PCDATA, then the content specification of $x_{G_{jk}}$ is the OR of the content specifications of $x_{G_{1_j}}$ and $x_{G_{2_k}}$.

6. If the content specification of both $x_{G_{1_j}}$ and $x_{G_{2_k}}$ consists of a set of sub-elements then also the content specification of $x_{G_{jk}}$ consists of a set of sub-elements. In particular, the set of sub-elements of $x_{G_{jk}}$ is obtained by considering the union of the sets of sub-elements of $x_{G_{1_j}}$ and $x_{G_{2_k}}$. If a synonymy $[x'_{1_j}, x'_{2_k}]$ holds in SD such that x'_{1_j} (resp., x'_{2_k}) is the local copy of a sub-element $x'_{G_{1_j}}$ (resp., $x'_{G_{2_k}}$) of $x_{G_{1_j}}$ (resp., $x_{G_{2_k}}$) then $x'_{G_{1_j}}$ and $x'_{G_{2_k}}$ are merged in $x'_{G_{jk}}$. The minimum (resp., maximum) cardinality of $x'_{G_{jk}}$ is set

[8] Clearly, when we mention the "same attributes" and "same sub-elements" we intend the attributes and the sub-elements in $\mathcal{T}_\mathcal{G}$ corresponding to the attributes and the sub-elements of x_{1_j} (resp., x_{2_k}).

Type of $x_{G_{1A}}$	Type of $x_{G_{2A}}$	Type of $x_{G_{12A}}$		Type of $x_{G_{1A}}$	Type of $x_{G_{2A}}$	Type of $x_{G_{12A}}$
CDATA	*CDATA*	*CDATA*		*CDATA*	*NMTOKEN*	*CDATA*
CDATA	*NMTOKENS*	*CDATA*		*ID*	*ID*	*ID*
IDREF	*IDREF*	*IDREF*		*IDREF*	*IDREFS*	*IDREFS*
IDREFS	*IDREF*	*IDREFS*		*IDREFS*	*IDREFS*	*IDREFS*
NMTOKEN	*CDATA*	*CDATA*		*NMTOKEN*	*NMTOKEN*	*NMTOKEN*
NMTOKEN	*NMTOKENS*	*NMTOKENS*		*NMTOKENS*	*CDATA*	*CDATA*
NMTOKENS	*NMTOKEN*	*NMTOKEN*		*NMTOKENS*	*NMTOKENS*	*NMTOKENS*
NOTATION	*NOTATION*	*NOTATION*		*NOTATION*	*ENTITY*	*NOTATION*
NOTATION	*ENTITIES*	*NOTATION*		*ENTITY*	*NOTATION*	*NOTATION*
ENTITY	*ENTITY*	*ENTITY*		*ENTITY*	*ENTITIES*	*ENTITIES*
ENTITIES	*NOTATION*	*NOTATION*		*ENTITIES*	*ENTITY*	*ENTITIES*
ENTITIES	*ENTITIES*	*ENTITIES*				

Table 2: Definition of the type of the attribute $x_{G_{12A}}$ from the types of the attributes $x_{G_{1A}}$ and $x_{G_{2A}}$

to the minimum (resp., the maximum) between the minimum (resp., the maximum) cardinalities of $x'_{G_{1_j}}$ and $x'_{G_{2_k}}$.

7. If the content specification of $x_{G_{1_j}}$ (resp., $x_{G_{2_k}}$) consists of a set of subelements and the content specification of $x_{G_{2_k}}$ (resp., $x_{G_{1_j}}$) consists of an OR of other content specifications then the content specification of $x_{G_{jk}}$ consists of the OR of the content specifications of $x_{G_{1_j}}$ and $x_{G_{2_k}}$.

8. If the content specification of both $x_{G_{1_j}}$ and $x_{G_{2_k}}$ consists of an OR of other content specifications then the content specification of $x_{G_{jk}}$ consists of the OR of the content specifications of $x_{G_{1_j}}$ and $x_{G_{2_k}}$.

9. The set of attributes of $x_{G_{jk}}$ is obtained from the union of the sets of attributes of $x_{G_{1_j}}$ and $x_{G_{2_k}}$. If there exist two attributes $x_{G_{1A}}$ and $x_{G_{2A}}$, such that a synonymy holds between the corresponding local copies, then they must be merged into an attribute $x_{G_{12A}}$; the name of $x_{G_{12A}}$ is the name of either $x_{G_{1A}}$ or $x_{G_{2A}}$; the type of $x_{G_{12A}}$ is determined from those of $x_{G_{1A}}$ and $x_{G_{2A}}$ by applying rules shown in Table 2[9]. The minimum (resp., maximum) cardinality of $x_{G_{12A}}$ is set to the minimum (resp., the maximum) between the minimum (resp., the maximum) cardinalities of $x_{G_{1A}}$ and $x_{G_{2A}}$.

After all synonymies have been examined, homonymies and second kind type conflicts must be analyzed. In particular:

– For each homonymy $[x_{1_j}, x_{2_k}]$ holding in HD, the name of either $x_{G_{1_j}}$ (i.e., the global duplicate of x_{1_j} in $\mathcal{T}_\mathcal{G}$) or $x_{G_{2_k}}$ (i.e., the global duplicate of x_{2_k} in $\mathcal{T}_\mathcal{G}$) must be modified.

[9] Remember that there exists a synonymy between two attributes if they are similar and their types are compatible. As a consequence, in Table 2, rules for determining the type of $x_{G_{12A}}$ are provided only for those cases for which types of $x_{G_{1A}}$ and $x_{G_{2A}}$ are compatible.

– For each second kind type conflict $[x_{1_j}, x_{2_k}]$ existing in $SKTCD$, the name of either $x_{G_{1_j}}$ (i.e., the global duplicate of x_{1_j} in $\mathcal{T}_{\mathcal{G}}$) or $x_{G_{2_k}}$ (i.e., the global duplicate of x_{2_k} in $\mathcal{T}_{\mathcal{G}}$) must be modified.

At the end of this phase, the global DTD $\mathcal{T}_{\mathcal{G}}$ has been completely derived and a set of mapping rules between each x-component of $\mathcal{T}_1$ (resp., $\mathcal{T}_2$) and each x-component of $\mathcal{T}_{\mathcal{G}}$ has been determined.

The second phase of the algorithm examines all element instances of the normalized document D_1 (resp., D_2) and, for each of them, generates a new element instance in D_G according to the mapping rules determined in the previous phase. Note that, in this phase, D_G conforms to the global DTD $\mathcal{T}_{\mathcal{G}}$ and similar concepts stored in D_1 and D_2 are represented by the same elements and attributes in D_G. Moreover, note that, since D_G has been derived by the *normalized* documents D_1 and D_2, it does not contain first kind type conflicts, whereas second kind type conflicts and homonymies are handled by the mapping rules.

The following theorem states the computational complexity of the integration activity; its proof can be found in the Appendix.

Theorem 30. Let D_1 and D_2 be two XML documents. Let n be the maximum between $|XCompSet(D_1)|$ and $|XCompSet(D_2)|$ and let N_{inst} be the maximum number of instances of D_1 and D_2. The worst case time complexity for integrating D_1 and D_2 is $O(n^4 + N_{inst})$. □

With regard to this result we observe that the dependency of the computational complexity on n^4 is mainly a theoretical result. Indeed, it derives from the necessity of applying the union operator on two sets of $O(n)$ sub-elements. Actually, in real situations, each element has a very limited number of sub-elements; as a consequence, the dependency of the computational complexity on n is generally quadratic.

Example 4. Consider the XML documents U_1 and U_2 shown in Figures 1 and 3, whose DTDs are illustrated in Figures 2 and 4. The integration algorithm receives U_1, U_2 and the flexibility level u against which the integration task must be performed. Assume the user specifies a flexibility level equal to 1. In this case, the Synonymy Dictionary, the Homonymy Dictionary, the First Kind Type Conflict Dictionary and the Second Kind Type Conflict Dictionary for U_1 and U_2 are:

$$\begin{aligned} SD = \{ & [University_{[U_1]}, University_{[U_2]}], [Professor_{[U_1]}, Professor_{[U_2]}], \\ & [Phone_{[U_1]}, Phone_{[U_2]}], [e\text{-}mail_{[U_1]}, e\text{-}mail_{[U_2]}], [Course_{[U_1]}, Course_{[U_2]}], \\ & [Student_{[U_1]}, Student_{[U_2]}], [Id_{[U_1]}, Identifier_{[U_2]}], [Name_{[U_1]}, Name_{[U_2]}], \\ & [Birthdate_{[U_1]}, Birthdate_{[U_2]}], [Teaches\_in_{[U_1]}, Teaches\_in_{[U_2]}], \\ & [Enrollment\_Year_{[U_1]}, Enrollment\_Year_{[U_2]}]\} \end{aligned}$$

$$HD = \emptyset$$

$$FKTCD = \{[Student\_Number_{[U_1]}, Student\_Number_{[U_2]}], \\ [Argument_{[U_1]}, Argument_{[U_2]}], [Date_{[U_1]}, Date_{[U_2]}], \\ [Mark_{[U_1]}, Grade_{[U_2]}]\}$$

$$SKTCD = \emptyset$$

Initially, U_1 and U_2 are normalized for solving all first kind type conflicts. As an example, the first kind type conflict between the attribute $Student\_Number_{[U_1]}$ of the element $Course_{[U_1]}$ and the element $Student\_Number_{[U_2]}$ is solved as follows. First the attribute $Student\_Number_{[U_1]}$ is transformed into the element:

```
<!ELEMENT Student_Number (#PCDATA)>
   <!ATTLIST Student_Number
     Identifier ID #REQUIRED
   >
```

then the attribute

```
Student_Number_Ref IDREF #REQUIRED
```

is added to $Course_{[U_1]}$. The attribute $Student\_Number_{[U_1]}$ is removed from $Course_{[U_1]}$ and the pair $[Student\_Number_{[U_1]}, Student\_Number_{[U_2]}]$ is moved from $FKTCD$ to SD. All the other first kind type conflicts are solved in an analogous way.

After U_1 and U_2 have been normalized, all their x-components are duplicated in the global DTD U_G. After this operation, U_G could contain some redundancies and/or ambiguities; these must be removed in order to obtain a final refined version of it.

The first step of this refinement phase examines all synonymies stored in SD. As an example, consider the synonymous elements $Professor_{[U_1]}$ and $Professor_{[U_2]}$; they must be merged in one single element $Professor_{[U_G]}$. The content specification of both $Professor_{[U_1]}$ and $Professor_{[U_2]}$ consists of a set of sub-elements; therefore, also the content specification of $Professor_{[U_G]}$ consists of a set of sub-elements, obtained from the union of the set of sub-elements of $Professor_{[U_1]}$ and $Professor_{[U_2]}$. Since the synonymy $[Phone_{[U_1]}, Phone_{[U_2]}]$ is stored in SD and $Phone_{[U_1]}$ (resp., $Phone_{[U_2]}$) belongs to the set of sub-elements of $Professor_{[U_1]}$ (resp., $Professor_{[U_2]}$), $Phone_{[U_1]}$ and $Phone_{[U_2]}$ are merged in a single sub-element $Phone_{[U_G]}$ of $Professor_{[U_G]}$. The minimum (resp., the maximum) cardinality of $Phone_{[U_G]}$ is set to the minimum (resp., the maximum) of the minimum (resp., the maximum) cardinalities of $Phone_{[U_1]}$ and $Phone_{[U_2]}$. All the other synonymies holding between sub-elements of $Professor_{[U_1]}$ and $Professor_{[U_2]}$ are handled in an analogous way.

The attributes of $Professor_{[U_G]}$ are obtained from the union of the attributes of $Professor_{[U_1]}$ and $Professor_{[U_2]}$. If two of these attributes are synonymous, they must be merged. As an example, consider the attributes $Identifier_{[U_1]}$,

```
<!ELEMENT University (Professor+, Department+, Student+, Course+)>
<!ELEMENT Professor (Phone+, e-mail*, Project*)>
   <!ATTLIST Professor Identifier ID #REQUIRED Name CDATA #REQUIRED
      Birthdate CDATA #IMPLIED Birthplace CDATA #IMPLIED
      Salary CDATA #IMPLIED Belongs_to IDREF #IMPLIED Teaches_in IDREFS #IMPLIED
      Papers IDREFS #IMPLIED
   >
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT e-mail (#PCDATA)>
<!ELEMENT Project (#PCDATA)>
   <!ATTLIST Project Starting_Date CDATA #IMPLIED
      Ending_Date CDATA #IMPLIED
   >
<!ELEMENT Department (EMPTY)>
   <!ATTLIST Department
      Identifier ID #REQUIRED Director CDATA #REQUIRED
      Floor CDATA #IMPLIED
   >
<!ELEMENT Course (EMPTY)|(Year?, Student_Number?, Argument*)>
   <!ATTLIST Course Identifier ID #REQUIRED
      Name CDATA #REQUIRED Student_Number_Ref IDREF #REQUIRED
      Argument_Ref IDREF #REQUIRED Duration CDATA #REQUIRED
      Attended_by IDREFS #IMPLIED Teached_by IDREF #REQUIRED
      Responsible IDREF #REQUIRED Propaedeutic_Courses IDREFS #IMPLIED
   >
<!ELEMENT Student_Number (#PCDATA)>
   <!ATTLIST Student_Number Identifier ID #REQUIRED >
<!ELEMENT Argument (#PCDATA)>
   <!ATTLIST Argument Identifier ID #REQUIRED >

<!ELEMENT Student (Test*, Address?, Thesis?)>
   <!ATTLIST Student Identifier ID #REQUIRED
      Name CDATA #REQUIRED Average_Mark CDATA #REQUIRED
      Birthdate CDATA #IMPLIED Birthplace CDATA #IMPLIED
      Enrollment_Year CDATA #IMPLIED Tutor CDATA #IMPLIED
      Attends IDREFS #IMPLIED Exams IDREFS #IMPLIED
      Attended_Courses IDREFS #IMPLIED
   >
<!ELEMENT Test>
   <!ATTLIST Test Identifier ID #REQUIRED
      Date_Ref IDREF #REQUIRED Mark_Ref IDREF #REQUIRED
      Argument_Ref IDREF #REQUIRED
   >
<!ELEMENT Paper (#PCDATA)>
   <!ATTLIST Paper Id ID #REQUIRED
      Authors IDREFS #REQUIRED Type (Journal|Conference) ''Journal''
      Volume CDATA #REQUIRED Pages CDATA #REQUIRED
   >
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT Thesis (Title, Topic+)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Topic (#PCDATA)>
<!ELEMENT Exam (Date, Mark)>
   <!ATTLIST Exam Id ID #REQUIRED
      Student IDREF #REQUIRED Course IDREF #REQUIRED
   >
<!ELEMENT Date (#PCDATA)>
   <!ATTLIST Date Identifier ID #REQUIRED >
<!ELEMENT Mark (#PCDATA)>
   <!ATTLIST Mark Identifier ID #REQUIRED >
```

Figure 6: The integrated DTD U_G

relative to $Professor_{[U_1]}$, and $Id_{[U_2]}$, relative to $Professor_{[U_2]}$; since they are synonymous, they must be merged into a single attribute of $Professor_{[U_G]}$ whose name is $Identifier_{[U_G]}$, whose type is ID and whose minimum (resp., maximum) cardinality is obtained by computing the minimum (resp., the maximum) between the minimum (resp., the maximum) cardinalities of $Identifier_{[U_1]}$ and $Id_{[U_2]}$. All the other synonymous attributes are merged analogously.

As for this pair of documents, no homonymy and no second kind type conflict have been found with a flexibility level equal to 1; therefore, no further action must be carried out on U_G. As a consequence, the integration activity terminates and the global DTD U_G, shown in Figure 6, is obtained.

□

6 Experiments

As it will be clear in Section 7, many integration approaches have been proposed in the literature. In order to provide a uniform evaluation of them, [6] proposed a catalogue of criteria and exploited them for comparing some of the most popular systems, i.e., *Autoplex* [1], *Automatch* [2], *COMA* [7], *Cupid* [22], `LSD` [8], `GLUE` [9], *SemInt* [19] and `SF` (*Similarity Flooding*) [24].

In our opinion this is a very interesting effort and we have decided to exploit the same criteria for testing the performances of our approach. This allowed us to obtain an objective and independent evaluation of it as well as to make a precise comparison between it and the other systems evaluated by [6].

In our experimental tests we have exploited a set of XML documents relative to various application contexts. More specifically, a first group of documents concerned the management of projects financed by European Union; a second group was relative to land and urban property registers; finally, a third group handled financial information. Such a variety of documents, derived from disparate application contexts, is justified by our desire to test the behaviour of our approach in many application environments.

Examined sources were characterized by the following properties, expressed according to the terminology and the measures of [6]:

- *number of documents*: we have considered 10 XML documents whose characteristics are reported in Table 3; this number of documents is quite similar to those considered by the authors of the other approaches for performing their evaluation; they are reported in Table 4 (see [6] for more details). From this table it is possible to see that the number of documents exploited for carrying out this kind of evaluation ranges from 5 to 24.

- *size of documents*: the size of the intensional component of the evaluated XML documents, i.e., the number of their elements and attributes, ranged from 16 to 75 (see Table 3); the average size is 40. The minimum, the maximum and the average size of the sources evaluated by the other approaches have been derived by [6] and are reported in Table 4[10]. An analysis of this table shows that the sizes of documents evaluated by our approach are quite close to the sizes of sources examined by the other systems. The size of

[10] The size of a relational source has been intended as the number of its relations and attributes.

	Doc. 1	*Doc. 2*	*Doc. 3*	*Doc. 4*	*Doc. 5*	*Doc. 6*	*Doc. 7*	*Doc. 8*	*Doc. 9*	*Doc. 10*
Max Depth	4	4	3	3	4	3	3	4	4	4
Number of x-components	64	75	16	40	44	25	26	46	37	28
Number of complex elements	14	14	9	8	6	7	6	9	6	7

Table 3: Some characteristics of the evaluated XML documents

System	*Tested document type*	*Number of documents*	*Minimum size of documents*	*Maximum size of documents*	*Average size of documents*
Our system	XML	10	16	75	40
Autoplex & Automatch	Relational	15	-	-	-
COMA	XML	5	40	145	77
CUPID	XML	2	40	54	47
LSD	XML	24	14	66	-
GLUE	XML	3	34	333	143
SemInt	Relational	10	6	260	57
SF	XML	18	5	22	12

Table 4: Characteristics of documents evaluated by the various approaches

test documents plays a relevant role because it influences the quality of synonymies; indeed, as mentioned in [6], the bigger the input documents are, the greater the search space for match candidates is and the lower the quality of obtained results will be.

In order to evaluate our approach and to compare its performances with those of the systems analyzed in [6], we have focused our attention on the synonymy extraction task.

All evaluation measures proposed in [6] and computed during our test campaign have been performed according to the following general framework:

– a set of experts has been asked to identify synonymies existing among involved documents;

– synonymies among the same documents have been determined by the approach to evaluate;

– the synonymies provided by the experts and those returned by the approach to test have been compared and evaluation measures have been computed.

In order to introduce the measures presented in [6] and computed in our experiments, it is necessary to consider the following definition:

Definition 31. Let A' be the set of synonymies provided by the experts and let C' be the set of synonymies returned by the approach to evaluate. The following sets can be defined:

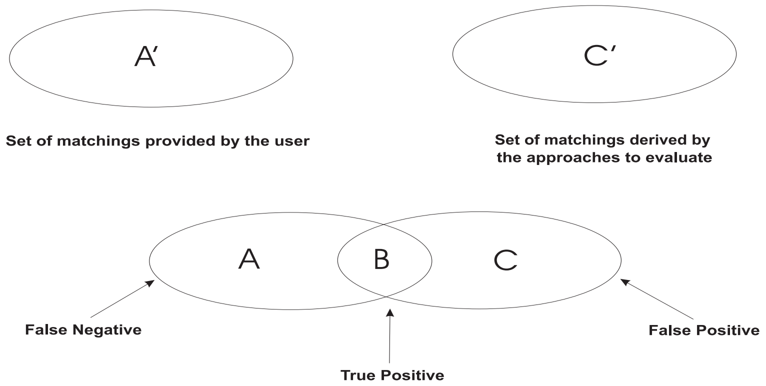

Figure 7: Explanation of Property Quality Metrics

- B (*True Positives*): this set consists of the synonymies automatically derived by the system that have been also recognized by the experts. It is possible to verify that $B = A' \cap C'$.

- A (*False Negatives*): this set consists of the synonymies provided by the experts that have not been derived by the system. A can be computed as $A = A' - B$.

- C (*False Positives*): this set consists of the synonymies derived by the system that have not been provided by the experts. C is defined as $C = C' - B$. $\square$

Figure 7 shows graphically how A, B and C can be defined from A' and C'. Two basic measures for evaluating the quality of an approach are:

- *Precision* (hereafter *Pre*), defined as:

$$Pre = \frac{|B|}{|B|+|C|} = \frac{|B|}{|C'|}$$

  which specifies the share of correct synonymies detected by the system among those it derived.

- *Recall* (hereafter *Rec*), defined as:

$$Rec = \frac{|B|}{|B|+|A|} = \frac{|B|}{|A'|}$$

  which indicates the share of correct synonymies detected by the system among those the experts provided.

Precision and Recall are typical measures of Information Retrieval techniques (see [34]). In [6] it is shown how they can be adapted to interscheme property derivation context. Both of them fall within the interval $[0, 1]$. Moreover, in the ideal case (i.e., when $|A| = |C| = 0$) they are both equal to 1; as a consequence, it is possible to state that the greater Precision (resp., Recall) is, the better an interscheme property derivation algorithm works.

As far as our approach is concerned, it is worth observing that the set C' of properties it derives varies with the flexibility level. As a consequence, C' depends on the flexibility level u and, in order to make this evident, we shall use the symbol $C'(u)$ instead of C'. The two sets A' and $C'(u)$ can be defined as:

$$A' = \{(x_{1_j}, x_{2_k}) | \ x_{1_j}(resp., x_{2_k}) \text{ is an x-component of } D_1 \text{ (resp., } D_2) \text{ and } \tau(x_{1_j}, x_{2_k}) = true\}$$

$$C'(u) = \{(x_{1_j}, x_{2_k}) | \ x_{1_j}(resp., x_{2_k}) \text{ is an x-component of } D_1 \text{ (resp., } D_2) \text{ and } synonymy(x_{1_j}, x_{2_k}, u) = true\}$$

where D_1 and D_2 are two involved XML documents, $\tau(x_{1_j}, x_{2_k})$ is a boolean function that returns *true* if the experts specified that a synonymy exists between x_{1_j} and x_{2_k} and $synonymy(x_{1_j}, x_{2_k}, u)$ has been defined in Section 4.2.

As for the evaluation of Precision and Recall relative to our algorithm, we argued that, due to its philosophy and intrinsic structure, an increase of the flexibility level should cause a decrease of Precision and an increase of Recall. This intuition is motivated by considering that $C'(u) \subseteq C'(u+1)$ and that $C'(u+1)$ is obtained from $C'(u)$ by adding some synonymies between concepts having a weaker similarity. This should cause $C'(u)$ to be more precise than $C'(u + 1)$; however, in this way, some valid synonymies could be added; for this reason $C'(u+1)$ could have a higher Recall w.r.t. $C'(u)$. In order to verify this intuition and, possibly, to quantify it, we have applied our approach on our test documents and we have computed Precision and Recall at various flexibility levels. Obtained results are shown in Table 5. These results confirmed our intuitions. Indeed, at the flexibility level 1, Recall increases of about 9% whereas Precision decreases of about 2% w.r.t. the flexibility level 0. As for flexibility levels greater than 1, we have verified that Recall constantly increases up to 1.00 whereas Precision slightly decreases at level 2 but, after this, it remains quite constant.

In our opinion such a result is extremely relevant; indeed, it allows us to conclude that, in formal situations, the right flexibility level is 0 whereas, in more informal contexts, the most suitable flexibility level is 2.

All experiments confirmed our original intuition about the trend of Precision and Recall in presence of variations of the flexibility level. In addition, values of Precision and Recall for increasing flexibility levels confirm observation of [6] about the behaviour of interscheme property derivation algorithms when the search space increases (see above); indeed, if flexibility level increases, the size

Flexibility Level	*Precision*	*Recall*
Level 0	0.98	0.88
Level 1	0.96	0.97
Level 2	0.95	0.99
Level 3	0.95	1.00

Table 5: Precision and Recall of our approach at various flexibility levels

System	*Precision*	*Recall*
Our system ($u = 0$)	0.98	0.88
Our system ($u = 1$)	0.96	0.97
Autoplex & Automatch	0.84	0.82
COMA	0.93	0.89
CUPID	–	–
LSD	~ 0.80	0.80
GLUE	~ 0.80	0.80
SemInt	0.78	0.86
SF	-	-

Table 6: Comparison of Precision and Recall between our system and various other ones already proposed in the literature

of considered neighborhoods increases, the "search space" grows and the overall quality of the properties decreases.

No further flexibility levels have been considered since the maximum depth of the evaluated documents is 4.

After this, we have compared the performances of X-Global w.r.t. the other ones evaluated in [6]. The results are reported in Table 6. From the analysis of this table we can observe that:

- At flexibility levels 0 and 1 X-Global shows the highest precision.

- At the flexibility level 0 the Recall of X-Global is quite high, even if COMA presents better performances. At the flexibility level 1 X-Global presents the highest Recall.

In our opinion, all the experiments we have conducted agree on determining that performances of X-Global are extremely satisfactory and promising. This fact is even more relevant if we consider that measures we have exploited had been already uniformly applied on a large variety of systems previously presented in the literature.

It is worth pointing out that, in all the experiments we have carried out, the validity of properties derived by X-Global has been verified by asking some experts to identify *true* properties relating the involved documents. This line of reasoning is widely accepted in the literature [6] and finds its motivations in the observation that the detection of semantic concept similarities involves also a heuristic, i.e. subjective, component; this makes it impossible to define a formal and general evaluation framework that do not consider human experts.

However, in some well assessed contexts, the set of properties relative to a group of sources could be already known and widely accepted. As an example, consider a set of relational databases of a company where the similarities and the differences among the concepts stored therein are determined by the experience, the domain knowledge and the day-by-day exploitation. In these cases, this set of properties could be exploited as the reference for an objective evaluation of our approach. Such an evaluation could be realized by translating involved information sources in XML documents and by applying X-Global on these sources. However, it is worth pointing out that, even in this context, the intervention of the human expert, for determining which properties returned by X-Global are valid, appears necessary.

7 Related Work

In the literature many approaches for performing interscheme property extraction and data source integration have been proposed. Many of them are capable of operating on a large variety of information source formats whereas some of them are specific for DTD's. However *X-Global* is not yet another approach for performing these tasks; indeed, it is characterized by some specific features which distinguish it from all the other methodologies previously presented in the literature.

The most characterizing feature of *X-Global*, which is not present in any other approach proposed in the past, is its capability to choose the flexibility level against which both the interscheme property extraction and the data source integration are carried out. The benefits of such a feature have been illustrated in the previous section; here, we remark that, if we focus on the possibility to choose the flexibility degree, any other approach previously proposed in the past can be seen as a particular case in which the flexibility degree is preliminarily fixed to a specific value.

Another remarkable feature of *X-Global* is its performance: indeed, as pointed out in the previous section, our system shows very good performances from both the Precision and the Recall point of view. These measures are "objective" in the sense that the framework adopted for computing them is the same as that exploited in [6] for determining the performances of various other systems already presented in the literature. Interestingly enough, the suitable choice of the flexibility level allows our approach to obtain the best Precision and the best Recall among all the other approaches mentioned in [6].

Finally, if we compare *X-Global* with learning methods proposed for inter-scheme property extraction and data source integration we can observe that they have been conceived for quite different application contexts. Indeed, learning methods need a training phase allowing to tune them to the specific application domain which they are operating on. Vice versa, *X-Global* is static and, consequently, it cannot adapt itself to the context where it is applied at a given moment. From this point of view, learning methods may obtain more accurate results; however, their training phase requires a computation effort and this makes them difficult to be applied in general contexts characterized by the presence of a large number of data sources to examine. Such a context is exactly that one specific for *X-Global*; indeed, as pointed out in the Introduction, our system has not weights and thresholds; therefore, it does not need a pre-processing phase and, consequently, it can operate on many information sources.

In the following we examine some of related approaches and highlight their similarities and differences w.r.t. *X-Global*.

In [18] the system *XClust* is presented whose purpose is the integration of XML data sources. More specifically, *XClust* determines the similarity degrees of a group of DTDs by considering not only the corresponding linguistic and structural information but also their semantics. This is derived by examining the neighborhoods of their elements; in particular a DTD is modeled as a tree and the neighborhood of an element consists of the set of its ancestors and descendants. The computation of the similarity degrees among the DTDs allows to group them into clusters; such a clustering activity is recursively applied on the DTDs of each generated cluster until a sufficiently small number of clusters is obtained.

It is possible to recognize some similarities between *X-Global* and *XClust*; in particular, *(i)* both of them have been specifically conceived for operating on XML data sources; *(ii)* both of them consider not only linguistic similarities but also semantic ones; *(iii)* both of them manage conflict resolution and both of them do not require a heavy human intervention; *(iv)* both of them are *rule-based* systems [30]. There are also several differences between them; more specifically, *(i)* in order to perform the integration activity, *XClust* requires the support of a hierarchical clustering whereas *X-Global* adopts schema matching techniques; *(ii) XClust* represents DTDs as trees; as a consequence, element neighborhoods are quite different from those constructed by *X-Global*; *(iii) XClust* exploits some weights and thresholds whereas *X-Global* 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 [25] the system *Rondo* is presented. It has been conceived for integrating and manipulating different data sources such as relational schemas or SQL views. It exploits a graph-based approach for modeling information sources and a set of high-level operators for matching obtained graphs. It uses the *Similarity Flooding Algorithm*, a graph-matching algorithm proposed in [24], to perform schema matching activity. Finally, it merges involved information sources according to three steps: *(i) Node Renaming*, that renames homonymous nodes; *(ii) Graph Union*, that creates a new graph by juxtaposing the two input ones; *(iii) Conflict Resolution*, that solves conflicts in the merged graph by deleting some of its edges.

There are important similarities between *Rondo* and our approach; indeed both of them are semi-automatic and exploit schema matching techniques. Moreover, both *Rondo* and *X-Global* manage conflict resolution and are *rule-based* systems. 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 *X-Global* is specialized for XML; *(ii) Rondo* models involved information sources as graphs whereas *X-Global* directly operates on XML Schemas; *(iii) Rondo* exploits a sophisticated technique (i.e., the Similarity Flooding Algorithm) for carrying out schema matching activities [24]; as a consequence, it obtains very precise results but is time-expensive and requires a heavy human feedback; on the contrary, *X-Global* is less sophisticated but is well suited when involved information sources are numerous and large.

The approach proposed in [4] carries out the integration of XML documents with the support of interscheme properties. In particular, an XML document is translated into a set of elements, called *x-classes*; this representation allows the derivation of synonymies, homonymies and type conflicts existing among concepts belonging to different sources; such a task is carried out by means of *clustering* techniques. Derived interscheme properties are exploited for carrying out the integration activity; this returns a global set of x-classes that is, in its turn, translated into a global XML document with the support of the user who can choose the structure of the final document.

X-Global has some similarities with the approach described in [4]. Indeed, both of them: *(i)* are *rule-based*; *(ii)* derive interscheme properties that are, then, exploited for carrying out the integration task; *(iii)* propose a type conflict resolution strategy. However, the two approaches present several differences; in particular: *(i)* The general philosophies underlying them are quite different; indeed, the approach proposed in [4] privileges result accuracy to the detriment of computational complexity; vice versa, the computations required by *X-Global* are less expensive. In our opinion, this last feature becomes crucial in a Web-oriented context where the number of sources to integrate is large. *(ii)* The intervention of the human expert required by the approach proposed in [4] is heavier than that needed by *X-Global*.

In [23] an approach for integrating information sources characterized by different representation formats (e.g., E/R, UML, XML) is proposed. It behaves as follows: first involved information sources are translated in a particular, auxiliary, graph-based formalism called HDM; then translated sources are integrated; the global source thus obtained is, finally, translated into one of the original formats.

The approach of [23] and *X-Global* differ mainly because the former has been conceived for allowing the integration of data sources characterized by a large variety of formats; vice versa, *X-Global* has been designed for integrating only XML documents. As a consequence, in order to integrate two XML documents, the approach of [23] needs to translate them into HDM and the global representation back into XML. Moreover, in [23] the mapping between two documents is carried out by means of suitable rules whereas *X-Global* exploits maximum weight matching techniques.

In [10] an XML-based integration approach, capable of handling various source formats, is presented. Involved sources are, first, translated into XML documents. After this, the DTDs of these documents are translated into a conceptual model named ORM/NIAM [14]. Finally, ORM/NIAM schemas are integrated for obtaining a global representation.

The approach of [10] and *X-Global* share the following features: *(i)* they operate on XML documents; *(ii)* they carry out a semantic integration; *(iii)* they handle type conflicts; *(iv)* if wrappers translating data sources from their own formats to XML documents are available, both of them can handle various source formats, even if they have been conceived mainly for integrating XML documents. The main differences between the approach of [10] and our own are: *(i)* the approach of [10] requires to translate the DTDs of involved XML documents in ORM/NIAM; vice versa, *X-Global* directly operates on XML documents; *(ii)* the global schema constructed by the approach of [10] is represented in ORM/NIAM whereas the integrated source returned by *X-Global* is represented in XML; *(iii)* the approach of [10] is quite complex and, therefore, can be applied with more difficulty w.r.t. *X-Global* when involved sources are numerous; *(iv)* [10] exploits clustering techniques to integrate two XML documents whereas *X-Global* is based on schema matching techniques.

[20] describes an approach performing the integration of data sources with different formats. Involved sources are first translated into a graph formalism named HDG. After this, all obtained HDG graphs are integrated; such a task is carried out by deriving semantic and structural relationships among objects belonging to different sources. Such a task is performed by applying a suitable set of rules. The global representation thus obtained is, finally, translated from HDG to XML.

The approach of [20] and our own are similar in that: *(i)* both of them are semantic; *(ii)* in both of them the integration is light, even if the approach of [20] requires a translation phase before the integration activity; *(iii)* both of them exploit a lexical dictionary, in particular WordNet; *(iv)* both of them are almost automatic and, finally, *(v)* both of them are *rule-based* and manage type conflicts. The two approaches present also various differences; in particular: *(i)* the approach of [20] has been conceived for integrating various information source formats whereas our own is specialized for XML; *(ii)* the approach described in [20] requires to translate input data sources into the HDG formalism before carrying out the integration task whereas *X-Global* does not need the translation of involved sources into a support formalism; *(iii)* the interscheme properties exploited for guiding the integration activity are different; in particular, [20] defines the relationship $\gg$, indicating if a concept is an hypernym or an ancestor of another, whereas *X-Global* exploits synonymies, homonymies and type conflicts and defines a suitable set of rules to handle synonymies and homonymies as well as to solve type conflicts.

In [32] the *DIXSE* (Data Integration for XML based on Schematic Knowledge) tool is presented, aiming at supporting the integration of a set of XML documents. In particular, documents to integrate are examined and a knowledge base relative to them is constructed; this is, then, exploited for carrying out the integration task. The construction of the knowledge base requires the support of the user who must collaborate for modeling the structure of the XML sources to integrate. As a consequence, in *DIXSE*, the "mapping activity" is carried out by applying a suitable set of rules.

DIXSE shares many features with *X-Global*. Indeed: *(i)* both of them are semantic; *(ii)* both of them operate on XML documents; *(iii)* both of them exploit structural and terminological relationships for carrying out the integration activity; *(iv)* both of them consider type conflicts. The main differences between *DIXSE* and *X-Global* reside in the technique for deriving interscheme relationships; indeed, *DIXSE* requires the support of the user whereas *X-Global* derives them automatically. In this way, properties obtained by applying *DIXSE* could be more precise than those returned by our technique but, when the number of sources to integrate is high, the effort required to the user might be particularly heavy.

In [8] 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 [9]. LSD requires quite a heavy support of the user during the initial phase, for carrying out the training activity; however, after this phase, no human intervention is required; after this, the system adopts *stacking techniques* [35, 33] to derive interscheme properties. Both LSD and *X-Global* operate mainly on XML sources. They differ especially in their purposes; indeed, LSD aims at deriving interscheme properties whereas *X-Global* 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 *X-Global* is *"rule-based"* [30]. Finally, LSD requires a heavy human intervention at the beginning and, then, is automatic; vice versa, *X-Global* does not need a pre-processing phase but requires the human intervention at the end for validating obtained results.

In [7] the authors propose COMA (COmbining MAtch), an interactive and iterative system for combining various schema matching techniques. The approach of COMA appears orthogonal to that of *X-Global*; in particular, our system 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, *X-Global* has been specifically conceived to handle XML documents. In addition, *X-Global* requires the user to specify only the *flexibility level*; vice versa, in COMA, the user must specify the *matching strategy* (i.e., the desired matchers to exploit and the modalities to combine their results).

In [16] *DEEP*, a system capable of integrating XML DTDs, is proposed. The architecture of *DEEP* consists of three major components: *(i)* A *DTD cluster*, that takes a set of DTDs as input and groups similar DTDs into clusters. To compute the "distance" among different DTDs, *DEEP* exploits *tree matching techniques* [21]; similar DTDs are clustered by applying a *hierarchical agglomerative clustering* method [31]. *(ii)* A *Schema Learner*, that derives an integrated schema from the original DTDs by applying a suitable set of rules. These are based on a tree grammar inference technique; in addition, sophisticated techniques are defined to induce hidden grammatical rules. *(iii)* An *Optimizer*, that "optimizes" the learned grammatical rules (e.g. it filters out the redundant ones).

DEEP and *X-Global* share some similarities; in particular, both of them: *(i)* operate only on XML documents, *(ii)* consider type conflict resolution, *(iii)* require a limited human intervention. The main differences between them are the following: *(i) DEEP* is a *learner-based* system, whereas *X-Global* is *rule-based*; *(ii)* due to the exploitation of sophisticated artificial intelligence techniques, *DEEP* is very precise but also very time expensive.

In [17] *X-Mapper*, a system capable of determining semantic mappings between two XML data sources, is presented. In order to "learn" semantic mappings, *X-Mapper* exploits the "constraints" associated with an XML document (e.g., data types, number of null values, and so on). In particular, for each pair of tags $\langle t_1, t_2 \rangle$ such that t_1 (resp., t_2) belongs to an XML document D_1 (resp., D_2), *X-Mapper* creates a *vector of features* describing the properties of the pair. The *DataSqueezer algorithm* [5] is then used to determine the pairs of tags having the maximum similarities; these pairs represent the final mappings. Interestingly enough, *X-Mapper* uses only *stand-alone* XML documents (i.e., XML documents without a DTD) to generate mappings; moreover, *X-Mapper* exploits machine learning techniques to improve the accuracy of results.

System	*Exploitation of an Intermediate Format*	*Data Source Typology*	*Conflict Resolution*	*Matching Technique*	*Human Feedback*	*Strategy*
X-Global	No	XML	Yes	Maximum Weight Matching	Light	Rule Based
X-Clust	No	XML	Yes	Clustering	Light	Rule Based
Rondo	Graph Based	Generic	Yes	Similarity Flooding	Heavy	Rule Based
Castano et al. [4]	X-Classes	XML	Yes	Clustering	Heavy	Rule Based
Mc Brien and Poulovassilis [23]	Graph Based (*HDM*)	Generic	Yes	Rule Mappings	Heavy	Rule Based
Dos Santos et al. [10]	Object Oriented (*ORM-NIAM*)	XML	Yes	Clustering	Heavy	Rule Based
Lim et Ng [20]	Graph Based (*HDG*)	Generic	Yes	Rule Mappings	Light	Rule Based
DIXSE	No	XML	Yes	Rule Mappings	Heavy	Rule Based
LSD-GLUE	No	Generic	Yes	Naive Bayes Stacking [33, 35]	Heavy	Learner Based
COMA	No	Generic	-	-	Heavy	-
DEEP	No	XML	Yes	Tree Matching[21] Hierarchical Clustering[31]	Light	Learner Based
X-Mapper	No	XML	Yes	DataSqueezer[5]	Light	Learner Based

Table 7: A comparison between some integration approaches and *X-Global*

X-Mapper and *X-Global* share some similarities; in particular, *(i)* both of them operate only on XML documents; *(ii)* both of them consider conflict resolution. The main differences existing among them are: *(i) X-Mapper* is a *learner-based* system whereas *X-Global* is *rule-based*; *(ii) X-Mapper* privileges accuracy to quickness.

A comparison between *X-Global* and the other systems we have mentioned above is reported in Table 7.

8 Conclusions

In this paper we have proposed *X-Global*, a system for the integration of XML documents. We have shown that *X-Global* is specialized for XML documents, is almost automatic, semantic and "light" and allows the choice of the "flexibility" 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 extend *X-Global* in such a way to handle XML Schemas. In addition, we plan to exploit it in various contexts typically benefiting of information source integration, such as Cooperative Information Systems, Data Warehousing, Semantic Query Processing and so on.

References

1. J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proc. of the International Conference on Cooperative Information Systems (CoopIS 2001)*, pages 108–122, Trento, Italy, 2001. Lecture Notes in Computer Science, Springer.
2. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. of the International Conference on Advanced Information Systems Engineering (CAiSE 2002)*, pages 452–466, Toronto, Canada, 2002. Lecture Notes in Computer Science, Springer.
3. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *Transactions on Data and Knowledge Engineering*, 13(2):277–297, 2001.
4. S. Castano, V. De Antonellis, A. Ferrara, and G. Kuruvilla. Ontology-based integration of heterogeneous XML datasources. In *Atti del Decimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD'02)*, pages 27–41, Portoferraio, Italy, 2002.
5. K.J Cios and L. Kurgan. Learning algorithms that generates compact rules. *Information Science - Special Issue on Soft Computing and Data Mining*, 2003 Forthcoming.
6. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proc. of the International Workshop on Web, Web-Services, and Database Systems*, pages 221–237, Erfurt, Germany, 2002. Lecture Notes in Computer Science, Springer.
7. 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.
8. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proc. of the International Conference on Management of Data (SIGMOD 2001)*, pages 509–520, Santa Barbara, California, USA, 2001. ACM Press.
9. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the Semantic Web. In *Proc. of the International Conference on World Wide Web (WWW 2002)*, pages 662–673, Honolulu, Hawaii, USA, 2002. ACM Press.
10. 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.
11. P. Fankhauser, M. Kracker, and E.J. Neuhold. Semantic vs. structural resemblance of classes. *ACM SIGMOD RECORD*, 20(4):59–63, 1991.
12. Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18:23–38, 1986.
13. M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: a system for extracting document type descriptors from XML documents. In *Proc. of the International Conference on Management of Data (SIGMOD 2000)*, pages 165–176, Dallas, Texas, United States, 2000. ACM Press.
14. 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.
15. D.S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.
16. E. Jeong and C. Hsu. View inference for heterogeneous XML information integration. *Journal of Intelligent Information Systems - Special Issue on "Web Intelligence"*, 20(1):81–99, 2003.
17. L. Kurgan, W. Swiercz, and K.J. Cios. Semantic mapping of XML tags using inductive Machine Learning. In *Proc. of the International Conference on Machine*

Learning and Applications (ICMLA'02), pages 99–109, Las Vegas, Nevada, USA, 2002.
18. M.L. Lee, L.H. Yang, W. Hsu, and X. Yang. XClust: clustering XML schemas for effective integration. In *Proc. of the International Conference on Information and Knowledge Management (CIKM 2002)*, pages 292–299, McLean, Virginia, USA, 2002. ACM Press.
19. W. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering*, 33(1):49–84, 2000.
20. S. Lim and Y. Ng. Semantic integration of semistructured data. In *Proc. of the International Symposium on Cooperative Database Systems and Applications (CODAS'01)*, pages 15–24, Beijing, China, 2001. IEEE Computer Society Press.
21. S.Y. Lu. A tree matching algorithm based on node splitting and merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):249–256, 1984.
22. 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.
23. P. McBrien and A. Poulovassilis. A semantic approach to integrating XML and structured data sources. In *Proc. of the International Conference on Advanced Information Systems Engineering (CAiSE'01)*, pages 330–345, Interlaken, Switzerland, 2001. Lecture Notes in Computer Science, Springer-Verlag.
24. 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 International Conference on Data Engineering (ICDE 2002)*, pages 117–128, San Jose, California, USA, 2002. IEEE Computer Society Press.
25. 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.
26. A.G. Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
27. S. Nestorov, S. Abiteboul, and R. Motwani. Extracting schema from semistructured data. In *Proc. of International Conference on Management of Data (SIGMOD'98)*, pages 295–306, Seattle, Washington, USA, 1998. ACM Press.
28. 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.
29. L. Palopoli, G. Terracina, and D. Ursino. A graph-based approach for extracting terminological properties of elements of XML documents. In *Proc. of the International Conference on Data Engineering (ICDE 2001)*, pages 330–337, Heidelberg, Germany, 2001. IEEE Computer Society.
30. E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
31. E. Rasmussen. *Clustering Algorithm.* Prentice-Hall, 1992.
32. 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-Verlag.
33. K.M. Ting and I.H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.
34. C. J. van Rijsbergen. *Information Retrieval.* Butterworth, 1979.
35. D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

Appendix: Proofs of Propositions and Theorems

Proposition 17

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

Proof

The construction of $XG(D, \mathcal{T})$ can be performed by exploiting the algorithm shown in Figure 8. In this algorithm, the procedure *ComputeSupp* derives a support array *Supp* having a component for each instance of an attribute of type "ID" in D. In particular, $Supp[j] = x_k$ if x_k is an x-component and j is the instance of an attribute of type "ID" of x_k.

The procedure *AddNode* receives a graph XG and an x-component x and adds a node representing x to the set $N(\mathcal{T})$ of nodes of XG.

The function *Parent* receives a DTD $\mathcal{T}$ and an x-component x_T and returns the x-component x_S which x_T is a sub-element of.

The function *isAttributeOf* receives two x-components x_S and x_T and a DTD $\mathcal{T}$ and returns *true* if x_T is defined as an attribute of x_S in $\mathcal{T}$, *false* otherwise.

The procedure *AddArc* receives a graph XG, two x-components x_S and x_T and an integer v and adds the arc $\langle x_S, x_T, v\rangle$ to the set $A(D, \mathcal{T})$ of arcs of XG.

The function *isSubelementOf* takes a DTD $\mathcal{T}$ and two x-components x_S and x_T as input and returns *true* if x_T is a sub-element of x_S in $\mathcal{T}$, *false* otherwise.

The function *isIDREFS* receives a DTD $\mathcal{T}$ and an attribute instance a_{inst} and returns *true* if a_{inst} is defined as an IDREF or IDREFS attribute in $\mathcal{T}$, *false* otherwise.

The function *xcomp* receives the instance of an element and returns the corresponding element.

The construction of *Supp* requires exactly one scan of D and, consequently, its computational complexity is $O(N_{inst})$.

The construction of $N(\mathcal{T})$ can be carried out by scanning $\mathcal{T}$ and creating a node of $N(\mathcal{T})$ for each encountered x-component. This activity costs $O(n)$.

The construction of $A(D, \mathcal{T})$ is performed by scanning $\mathcal{T}$ and, for each encountered x-component x_T, by verifying if it is an attribute or a sub-element of an x-component x_S. In the former case the arc $\langle N_S, N_T, 0\rangle$ is added to $A(D, \mathcal{T})$ whereas, in the latter case, the arc $\langle N_S, N_T, 1\rangle$ is inserted into $A(D, \mathcal{T})$, if not already present. The computational cost of all the operations described above is $O(n)$.

In order to complete the construction of $A(D, \mathcal{T})$, it is necessary to scan D and, for each IDREF(S) attribute instance, to examine its value for determining

```
Algorithm for the construction of XG(D,T) from D and T
Input: an XML document D, a DTD T.
Output: an X-Dist-Graph XG relative to D and T

var:
    Supp: an Array;
    XG: a Graph;
    x_S, x_T: an x-component;
    e_inst: an element instance;
    a_inst: an attribute instance;
    IS: a set of values;
begin
    ComputeSupp(D, T, Supp);
    XG := ∅;
    for each x-component x_S ∈ T do
        AddNode(XG, x_S);
    for each x-component x_T ∈ T do begin
        x_S := Parent(T, x_T);
        if isAttributeOf(T, x_S, x_T) then
            AddArc(XG, x_S, x_T, 0);
        else if isSubElementOf(T, x_S, x_T) then
            AddArc(XG, x_S, x_T, 1);
    end;
    for each element_instance e_inst ∈ D do
        for each attribute_instance a_inst ∈ e_inst do
            if isIDREFS(T, a_inst) then begin
                x_S := xcomp(e_inst);
                IS := getValues(a_inst);
                for each ref ∈ IS do begin
                    x_T := Supp[ref];
                    AddArc(XG, x_S, x_T, 1);
                end;
            end;
end.
```

Figure 8: The algorithm for constructing $XG(D, \mathcal{T})$

the element instances it refers to and, for each of them, to find the associated x-component. Let i_s be one of these IDREF(S) attribute instance and let x_S be the element whose instance contains i_s; let x_{T_i} be one of the elements having at least one instance referred by i_s; then, the arc $\langle N_S, N_{T_i}, 1\rangle$ is added to $A(D, \mathcal{T})$, if not already present. As for the computational complexity of these tasks, we observe that: *(i)* the maximum number of IDREFS attribute instances is $O(N_{inst})$; *(ii)* each of the IDREFS instances could refer to at most $O(N_{inst})$ instances; *(iii)* determining the x-component referred by an instance of an IDREF(S) attribute has a constant cost, because each index of *Supp* is exactly the instance of an "ID" attribute and the corresponding cell of *Supp* contains the x-component the "ID" attribute belongs to; *(iv)* determining all the x-components referred by an instance of an IDREFS attribute costs $O(N_{inst})$. All these observations allow us to conclude that the computational cost for completing the construction of $A(D, \mathcal{T})$ is $O(N_{inst}^2)$.

Analogously, the theoretical number of arcs in $XG(D, \mathcal{T})$ is $O(n^2)$; such a quadratic dependency on n arises because each IDREFS attribute might refer to instances of several element types. However, in real situations, the number of *distinct* element types whose instances are referred by an IDREFS attribute can be limited by a constant (generally small) value. As a consequence, the out-degree of each arc of the graph can be considered limited by a constant value; therefore, in real situations, the number of arcs in $XG(D, \mathcal{T})$ is generally $O(n)$.

The reasoning described previously allows us to conclude that the worst case time complexity for constructing $XG(D, \mathcal{T})$ is $O(max\{n, N_{inst}^2\})$. □

Theorem 18

Let $XG(D, \mathcal{T})$ be the XS-Graph associated with an XML document D and the corresponding DTD $\mathcal{T}$ and let n be the number of x-components of D. The worst case time complexity for computing the set $\{neighborhood(x_i, j) | x_i \in XCompSet(D), 0 \leq j \leq n\}$ is $O(n^3)$.

Proof

In order to construct the set $\{neighborhood(x_i, j) | x_i \in XCompSet(D), 0 \leq j \leq n\}$, it is necessary to compute $CC(x_S, x_T)$ for each pair (x_S, x_T) such that both x_S and x_T are x-components of D. $CC(x_S, x_T)$ can be computed by determining the length of the shortest path between the nodes corresponding to x_S and x_T in $XG(D, \mathcal{T})$. The computation of the length of all shortest paths in a graph costs $O(n^3)$ which is, therefore, the cost for constructing the set $\{neighborhood(x_i, j) | x_i \in XCompSet(D), 0 \leq j \leq n\}$. □

Theorem 21

Let D_1 and D_2 be two XML documents; let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2); finally, let n be the maximum between $|XCompSet(D_1)|$ and $|XCompSet(D_2)|$. If x_{1_j} and x_{2_k} are not similar with a flexibility level equal to n, then they are dissimilar.

Proof

Immediate, by observing that, at the flexibility level n, our approach is considering *ext-neighborhood*(x_{1_j}, n) and *ext-neighborhood*(x_{2_k}, n) that contain all x-components of D_1 and D_2; therefore, the examination of a further flexibility level would return the same result because no new x-component could be added to either *ext-neighborhood*(x_{1_j}, n) or *ext-neighborhood*(x_{2_k}, n) and, consequently, no change could be possible w.r.t. level n. □

Theorem 22

Let D_1 and D_2 be two XML documents. Let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be an integer greater than or equal to 0. Finally, let p be the maximum between the cardinalities of *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u). The computational cost for evaluating if *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u) are similar is $O(p^3)$.

Proof

Immediate by taking into account that the computation of the maximum weight matching on a bipartite graph having $O(p)$ nodes costs $O(p^3)$ [12]. □

Theorem 23

Let D_1 and D_2 be two XML documents. Let x_{1_j} (resp., x_{2_k}) be an x-component of D_1 (resp., D_2). Let u be the selected flexibility level. Finally, let p be the maximum between the cardinalities of *ext-neighborhood*(x_{1_j}, u) and *ext-neighborhood*(x_{2_k}, u). The worst case time complexity for computing $similar(x_{1_j}, x_{2_k}, u)$ is $O((u+1) \times p^3)$.

Proof

In the worst case, in order to verify if a similarity holds between x_{1_j} and x_{2_k} with a flexibility level equal to u, it is necessary to compute the similarity of *ext-neighborhood*(x_{1_j}, v) and *ext-neighborhood*(x_{2_k}, v) for each v such that $0 \leq v \leq u$. By Theorem 22 each of these computations costs $O(p^3)$; therefore, the total computational complexity is $O((u+1) \times p^3)$.

Theorem 30

Let D_1 and D_2 be two XML documents. Let n be the maximum between $|XCompSet(D_1)|$ and $|XCompSet(D_2)|$ and let N_{inst} be the maximum number of instances of D_1 and D_2. The worst case time complexity for integrating D_1 and D_2 is $O(n^4 + N_{inst})$.

Proof

The integration activity consists of the following tasks:

- *Resolution of first kind type conflicts*; since the maximum number of first kind type conflicts is $O(n^2)$ and the resolution of each of them requires a constant time, the overall cost of this task is $O(n^2)$.
- *Examination of synonymies stored in SD*; the maximum number of synonymies that might be analyzed is $O(n^2)$. Each synonymy resolution implies the merge of two x-components. If the two x-components are attributes their merge can be carried out in a constant time; vice versa, if they are elements, operations described in Points 1 to 9 of the algorithm described in Section 5 must be executed. In particular, Points 1 to 5, as well as Points 7 and 8, can be performed in a constant time. Point 6 (resp., Point 9) implies the application of a union operator on two sets of sub-elements (resp., attributes) each having dimension $O(n)$; this costs $O(n^2)$. Thus, the overall cost of analyzing SD is $O(n^4)$.
- *Resolution of homonymies and second kind type conflicts*; a reasoning analogous to that illustrated for the first kind type conflicts allows us to conclude that this task can be performed in $O(n^2)$.

- *Merge of instances*; this task can be carried out in $O(N_{inst})$.

From this analysis we can conclude that the worst case time complexity of our integration algorithm is $O(n^4 + N_{inst})$. □