Semantic aware matching and mapping for XML schemas

Tadeusz Pankowski
Institute of Control and Information Engineering
Poznań University of Technology, Poland
tadeusz.pankowski@put.poznan.pl

ABSTRACT: *Discovering matches and mappings between heterogeneous, independently designed data sources, is a challenging issue in data exchange and data integration. To deal with the problem we enrich XML schemas with semantic information from a domain ontology by annotating the schema. In this paper we discuss how the annotations can be used to establish matching and mapping formulas between XML schemas automatically and propose a set of rules to infer these formulas.*

Keywords: XML Schemas, Data sources, Data mapping, Data integration, Domain ontology, Semantic information

Received: 16 September 2009, Revised 11 November, Accepted 18 December 2009

1. Introduction

Establishing of schema matching and schema mapping are two important issues in developing data integration and data exchange systems, especially when schemas evolve and the data sources are considered in dynamic P2P data integration systems [15]. In this paper we discuss the problem of automatic creation of schema matching based on XML schema annotations into a domain ontology, and on the ontology itself. Next, the discovered matches are used to generate schema mappings.

Annotations are commonly used to enrich semantics of XML schemas [5, 20]. Schema matching techniques received a great deal of attention and were reported in many papers and surveys ([18, 10]). Matches create a key input to the creation of schema mappings. A formal foundation of mapping specification for relational data was proposed in [9]. An adaptation of this concept to XML data integration was discussed in [3, 13].

The contribution of this paper is the following: (1) we discuss annotation of XML schema as an annotation of its labels and (generalized) edges, into an OWL Lite domain ontology; (2) we define a language (based on tree-patterns) to express matches and mappings between schemas; (3) we propose inference rules to generate semantically meaningful matches based on annotations.

The paper is organized as follows: In Section 2 we discuss and illustrate by examples the ideas underlying the research. In Section 3, XML schemas and tree-pattern formulas are defined. XML schema annotation is discussed in Section 4, and rules for inferring schema matching are proposed in Section 5. In Section 6 we illustrate application of the method on an example. Section 7 concludes the paper.

2 Motivation and basic concepts

2.1 Basic concepts of annotation

To explain our ideas of generating schema matches and schema mappings from annotation, consider two sample XML schema trees (representing DTDs) in Figure 1. Text-valued variables placed in the figure are used to denote correspondences between text values of schema instances. Dotted arrows denote some (generalized) *edges* over DTDs chosen for the annotations. An edge can start with a label and end with: (a) its child, e.g. (*paper, conf*); (b) itself (a loop), e.g. (*paper*, *.paper*); (c) its descendant, e.g. (*pub, //year*); (d) its parent, e.g. (*writer, ..pub*); (e) its sibling, e.g. (*conf, ../year*).

Annotation of a schema concerns some labels and edges of the schema. The constructs are annotated with some terms from an OWL Lite ontology [12, 4]. These terms are: *class names* (*c*), and *property names* (*p*). Among *properties* we distinguish:

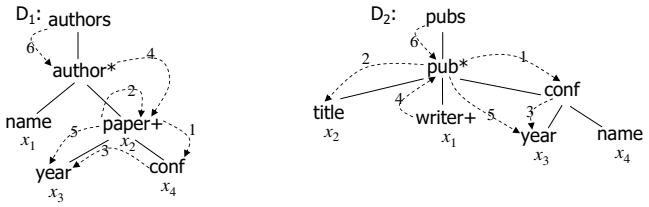

Figure 1. Sample schemas (DTDs) D_1 and D_2, and generalized edges (dotted arrows) chosen for the annotation

object properties that are triples of the form (*c, p, ć*), where *c* is the domain, *p* is the name, and *ć* is the range of the property, and *datatype properties*, i.e. triples of the form (*c*, *p*, *String*), where *c* is the domain, *p* is the name, and String is the range of the property.

1. *Annotation of labels*. A label l is annotated (by a function λ) with a set of OWL class names. Then c ∈ λ(*l*) means that *c* is a class name annotating *l*. For example: *Paper* ∈ λ(*paper*) (for D_1), and *Publication* ∈ λ(*pub*) (for D_2).
2. *Annotation of edges*. An edge is a pair of the form (*l*, *θl´*, where *l* and *l´* are labels and *θ* is an operator indicating how *l´* can be reached from *l*. An edge is annotated (by a function δ) with a set of OWL property names. For example: *Author of* ∈ δ(*author, paper*) (for D_1), *Presented at* ∈ δ(*pub, conf*) (for D_2); *Title of paper* ∈ δ(*paper, .paper*) (for D_1), *Title of publication* ∈ δ(*pub, title*) (for D_2), *Participant of* ∈ δ(*author, //conf*) (for D_1), and *Year of conference* ∈ δ(*conf, ../year*) (for D_1).

2.2 Similarity of names and compatibility of labels and edges

The annotating ontology O is a common domain ontology used by information engineers to annotate schemas. In general, an ontology is based on description logic (DL) [4] and is defined by means of a language based on DL OWL [12] or RDF(S). Then the DL inference rules, based on the solid semantics, are used to reason about concepts and assertions [17]. However, from our point of view semantic relations in DL, such as *subsumption* (⊑) and *equivalence* (≡), are often too restrictive. In many ontology-based applications some weaker semantic relations are assumed [7, 19]. We will use the *similarity* relation (≈), that holds between *similar* terms, for example between names from classes which have a common superclass. For example, *Author* ≈ *W riter, P aper* ≈ *P ublication*, *Author of* ≈ *inverseOf(Written by)*, etc. We assume that the set of axioms in a domain ontology O consists of strong relationships, formed using subsumption and equivalence, as well as from weak relationships constructed by means of ≈. Note that $A \equiv B$ implies $A \approx B$, and that ≈ is reflexive, symmetric but, in general, not transitive.

The crucial problem is establishing of a *compatibility* (or *matching* relation) between labels of two schemas. Intuitively, a label *l* from *D is compatible with* a label *l´* from *D´* if for any instance *I´* of *D´* there is an instance I´ of *D´* such that a nonempty part of *I* corresponds to a nonempty part of *I´*, and some additional semantic conditions are satisfied. The correspondence is specified by a *matching formula* that has a form of equivalence between two tree-pattern formulas (TPFs) [3, 13] over *D* and *D´*, respectively. For example, a matching formula between author and writer is (the universal quantification of variables is assumed):

$$//author[name[. = x_1]] \Leftrightarrow //writer[. = x_1],$$

thus the relation *author ~ writer* is the consequence of this formula. In this case the compatibility relation, *author ~ writer* between labels, coincides with the similarity relation *Author* ≈ *Writer* in the ontology. However, in general, such coincidences are not necessary. For example, it is easily seen that subtrees rooted in author and in pub are compatible, even though class names *Author* ∈ λ(*author*) and *Publication* ∈ λ(*pub*) are not similar, however we are able to define a matching formula between subtrees rooted in these labels. To enrich the semantic relationship between schemas, we additionally take into account semantics of edges leading from the root label of the subtree to its subtrees.

2.3 Matching and mapping of XML schemas

Matching formulas specify how the data from instances of compatible (parts of) schemas correspond to each other. In such specification the descendent operator, //, may occur in both sides of the equality, for example (Figure 1):

$$//paper[. = x_2, year[. = x_3]] \Leftrightarrow //pub[title[. = x_2], //year[. = x_3]] \quad (1)$$

A *mapping formula* is an implication created from a matching formula, where there is not any occurrence of in the right-hand side. The right-hand side is used to create the target instance for the set of variable valuations determined by the left-hand side of the mapping formula. Thus, the structure of the target must be defined unambiguously. To create the target instance

we can apply *chasing procedure* [1, 22]. In this procedure Skolem functions may be used [8, 14]. Thus, when we want to use the formula (1) to specify a mapping (transformation) from the left-hand side to a document with the root in pub, we must rewrite it (universal quantification of x_2 and x_3 is omitted, and result is an invented new root label):

$$//paper[. = x_2, year[. = x_3]] \Rightarrow x_1,x_4(/result[pub[title[. = x_2], writer[. = x_1], conf[year[. = x_3], name[. = x_4]]]]). \quad (2)$$

Semantics of TPFs is defined as a set of variable valuations for which the formula is satisfied on an instance I [16]. For example, the formula $//paper[. = x_2, year = x_3]$ is satisfied on the instance I_1 for valuation ω_1, such that $\omega_1(x_2)$= *Mapping* XML, $\omega_1(x_3) = 2009$.

In general, an equivalence, like (1), is satisfied for a pair (I_1, I_2) of instances, if the set of valuations for which the left-hand side is satisfied on I_1, is equal to the set of valuations for which the right-hand side is satisfied on I_2.

3. XML schemas and tree-pattern formulas

We assume that XML schemas are defined by simplified non-recursive DTDs (*Document Type Definition*), where attributes are represented by text nodes, and each regular expression has a simple form (like in [2]). As has been shown in [6, 11] such DTDs are very common in practice and cover about 70% of real-world situations. Let Σ be a set of *labels*, Text be a symbol denoting text values, and *Str* be a set of values of type *String*.

Definition 3.1 (*DTD*) *A tuple D =*(*top, Lab,* `Text`,ρ) *is a DTD, if Lab* ⊆ Σ*, top* ∈ *Lab is the root (the outermost) label, and* ρ *is a function assigning simple regular expressions, e, over Lab*\{*top*} ∪ {`Text`} *to labels,*

$$e ::= \texttt{Text} \mid l \mid l? \mid l^+ \mid l^* \mid e\ e,$$

where: l ∈ *Lab*\{*top*}*, and each label from Lab, except for top, occurs at most once and exactly in one regular expression.*

A label *l* ∈ *Lab* is: a *terminal* label, if ρ(*l*) = `Text`; a *mixed* label, if both `Text` and at least one label from *Lab* occur in ρ(*l*); a *non-terminal* label, if `Text` does not appear in ρ(*l*).

Definition 3.2 *A tree-pattern formula (TPF) over a set* Σ *of labels and a set x of variables is an expression with the syntax*

$$E ::= (. = x) \mid \theta l[E] \mid E, ..., E,$$
$$\theta ::= \varepsilon \mid . \mid // \mid .. \mid ../,$$

where l ∈ Σ*, x* ∈ *x,* ε *denotes an empty operator.*

The meaning of operators in TPFs, i.e. the dot operator (.) and θ*l*, in a context node *n* coincides with the semantics of XPath [21], i.e.:

.	:	abbreviates **`self`** :: *node*( ) and selects the context node *n*;
l	:	abbreviates **`child`** :: *l* and selects *l* child of *n*;
.l	:	abbreviates **`self`** :: *l* and selects *n* if its label is *l*;
//l	:	abbreviates **`descendant-or-self`**:: *l* and selects the node labeled *l* from the set consisting of *n*, all its children and all its descendants;
..l	:	abbreviates ***parent*** :: *l* and selects the parent of *n* if its label is *l*;
../l	:	abbreviates ***child*** :: *l*[***parent*** :: *node*( )] and selects the *l* child of the parent of *n*, i.e. an *l* sibling of *n*.

4. Schema annotation

An XML schema *D can be annotated in a domain ontol*ogy *O*. The annotation process assigns names of three OWL categories [12, 4]: *class names*, *object property names*, and *datatype property names* to schema constructs (labels and edges in the schema).

An annotation graph G_D for a DTD D, is a pair *(Lab, Edg)*, where *Lab* is the set of labels from D, and *Edg* is a set of edges of the form $(l, \theta l')$; $l, l' \in Lab$, $\theta \in \{\varepsilon, ., //, ../, ..\}$. For an edge $(l, \theta l')$, we say that begins with l, ends with l', and operator θ determines the way in which l' is reachable from l.

Among edges we distinguish *non-terminal*, *mixed*, and *terminal* edges, according to the kind of the ending label l'.

Let *CNames*, *OP Names* and *DT P Names* be sets of, respectively, class names, object property names and datatype property names in O.

Definition 4.1 *An annotation of D is a tuple* $Ann(G_D) = (\lambda, \delta)$*, where:*

1. $\lambda : Lab \rightarrow 2^{CNames}$ *– a function annotating labels with sets of class names;*
2. $\delta : Edg \rightarrow 2^{OP\ Names \cup DTP\ Names}$ *– a function annotating edges with sets of property names.*

Example 4.1 *Annotations of some labels from* D_1 *and* D_2 *are given in Table 1. Annotations of edges with object property names are listed in Table* 2*, and annotations of edges with datatype property names are given in Table 3.*

Note that some object and datatype properties may have equal names, (e.g. *Presented at* in Table 2 and in Table 3). It means that the union *OP Names* $\cup$ *DT P Names* in Definition 4.1 is understood as the *disjoint union*.

DTD	*Label*	*Class name*
D_1	*paper*	*Paper*
D_1	*paper*	*Title of paper*
D_1	*year*	*Year of publication*
D_1	*conf*	*Name of conference*
D_1	*conf*	*Conference*
. . .	. . .	. . .
D_2	*pubs*	*Publications*
D_2	*pub*	*Publication*
D_2	*title*	*Title of publication*
D_2	*writer*	*Writer*
D_2	*writer*	*Name of writer*
. . .	. . .	. . .

Table 1. Annotations of labels from D_1 *and* D_2

	DTD	*Edge*	*Property name*
(E1)	D_1	(*authors, author*)	*Contains*
(E2)	D_1	(*author, paper*)	*Author of*
(E3)	D_1	(*author, //conf*)	*Contributed to*
(E4)	D_1	(*paper, conf*)	*Presented at*
(E5)	D_2	(*pubs, pub*)	*Contains*
(E6)	D_2	(*pub, conf*)	*Presented at*
. . .	. . .	. . .	. . .

Table 2. Annotations of edges with object property names

	DTD	*Edge*	*Property name*
(E7)	D_1	*(paper, conf)*	*Presented at*
(E8)	D_2	*(pub, writer)*	*Written by*
(E9)	D_2	*(pub, //year)*	*Year of publication*
(E10)	D_2	*(writer, .writer)*	*Name of writer*
(E11)	D_2	*(pub,//name)*	*Presented at conference*
. . .	. . .	. . .	. . .

Table 3. Annotations of edges with datatype property names

5. Schema matching

5.1 Compatibility relation between labels and edges

Definition 5.1 (*Compatibility relation between labels*) *Two labels l and l′ from, respectively, schemas D and D′ are compatible, denoted*

$$l \sim l',$$

iff a subtree rooted in l can be matched with a subtree rooted in l′, i.e. there exists the following matching formula between D and D′

$$\forall x(//l[E(x)] \Leftrightarrow //l'[E'(x)]), \tag{3}$$

where //l[E(x)] and //l′[E′(x)] are TPFs over D and D′, respectively.

Definition 5.2 (*Satisfaction of a matching formula*) *A matching formula* (3) *is satisfied for schemas D and D′, iff for any instance I of D there exists an instance I′ of D′, such that the equality holds*

$$[\![//l[E(x)](I)]\!] = [\![//l'[E'(x)](I')]\!],$$

where $[\![E(x)(I)]\!]$ *is the value of* $E(x)$ *on the instance I. This value is the set of valuations of variables in x, for which the formula E(x) is true in I, i.e.*

$$[\![E(x)(I)]\!] = \{\omega \in [x \rightarrow String] \mid I \models E(\omega(x))\}.$$

Further on the universal quantification in matching formulas will be omitted.

Definition 5.3 *(Compatibility relation between edges) Two edges,* $(l_1, \theta l_2)$ *and* $(l'_1, \theta' l'_2)$*, are compatible, denoted*

$$(l_1, \theta l_2) \sim (l'_1, \theta' l'_2),$$

iff one of the following conditions holds:

1. *property names assigned to edges are similar, class names assigned to predecessors of these edges are similar, and the labels in the successors of the edges are compatible, i.e.*

$$\delta(l_1, l_2) \approx \delta(l'_1, l'_2) \wedge \lambda(l_1) \approx \lambda(l'_1) \wedge l_2 \sim l'_2;$$

2. *the object property name assigned to both edges is Contains, and the labels in the successors of the edges are compatible, i.e.*

$$\delta(l_1, l_2) \approx \delta(l'_1, l'_2) = Contains \wedge l_2 \sim l'_2.$$

Additionally,

$$(l_1, l_2) \approx inverseOf(l'_1, l'_2) = (l'_1, l'_2)^{-1} \text{ iff } (l_1, l_2) \approx (l'_2, l'_1).$$

5.2 Inferring semantically meaningful matching rules

Semantically meaningful matches are equivalences obtained by the following rules.

(R1) *Matching determined by mixed or terminal labels.* If l and l' are terminal labels, and $\lambda(l) \approx \lambda(l')$, then

$$//l[. = x] \Leftrightarrow //l'[. = x]$$

(R2) *Matching determined by compatible edges.*

If $(l_1, \theta l_2) \sim (l'_1, \theta' l'_2)$, and $//l_2[E] \Leftrightarrow //\theta_2[E']$, then

$$//l_1[\theta l_2[E]] \Leftrightarrow //l'_1[\theta' l'_2[E']].$$

(R3) *Additivity.*

If $//l[E_1] \Leftrightarrow //l'[E'_1]$, and $//l[E_2] \Leftrightarrow //l'[E'_2]$, then

$$//l[E_1, E_2] \Leftrightarrow //l'[E'_1, E'_2].$$

(R4) *Matching determined by inverse of edges.*

If $(l_1, l_2) \sim inverseOf(l'_1, l'_2)$ and
$//l_1[E_1] \Leftrightarrow //l'_2[E'_2]$ and $//l_2[E_2] \Leftrightarrow //l'_1[E'_1]$ then
$//l_1[E_1, l_2[E_2]] \Leftrightarrow //l'_1[E'_1, l'_2[E'_2]]$.

(R5) *Nesting for matches.* If $(l'_1, \theta' l'_2)$ is an element edge,

$$//l_1[\theta l_2[E_1]] \Leftrightarrow //l'_1[E'], \text{ and } //l_2[E_2] \Leftrightarrow //l'_2[E'_2],$$

then

$$//l_1[\theta l_2[E_1, E_2]] \Leftrightarrow //l'_1[E'_1, \theta' l'_2[E'_2]].$$

Example

Consider two sample XML schema trees, D_1 and D_2 in Figure 1. Variables in Figure 1 are intended to denote correspondences between text values of instances of these trees (note that *paper* is the label of mixed element). The goal is to discover those correspondences based on annotations of XML schemas. Annotations of D_1 and D_2 are listed in Tables 1 – 3. Axioms of the annotating ontology are given in Table 4.

(A1)	*Author ≈ W riter*
(A2)	*P aper ≈ P ublication*
(A3)	*Author of ≈ inverseOf(Written by)*
(A4)	*Name of author ≈ Name of writer*
(A5)	*Title of paper ≈ Title of publication*

Table 4. Axioms in the annotating ontology

D_1	D_2	*Matching formula*
name ~ writer		*//name*[. = x_1] ⇔ *//writer*[. = x_1]
paper ~ title		*//paper*[. = x_2] ⇔ *//title*[. = x_2]
year ~ year		*//year*[. = x_3] ⇔ *//year*[. = x_3]
conf ~ name		*//conf*[. = x_4] ⇔ *//name*[. = x_4]

Table 5. Compatibility of terminal labels

Compatibility relation on terminal labels is listed in Table 5. The second entry follows from the similarity between class names assigned to labels (see axiom (A5) in Table 4), i.e. λ*(paper)* ∋ *Title of paper ≈ Title of publication* ∈ λ*(title)*. Compatibility between terminal edges is given in Table 6.

Matching formulas for edges annotated with datatype property names listed in Table 6, are given in Table 7. They all are created in force of rule (R2).

Matching formulas in Table 8 are created as follows:

- *M7* = *R*3(*M*1,*R*3(*M*2,*M*4)),
- *M8* = *R*3(*M*3,*M*5),

	D_1	D_2
(1)	*(paper, .paper)*	*~ (pub, title)*
(2)	*(paper, year)*	*~ (pub, //year)*
(3)	*(conf, ../year)*	*~ (conf, year)*
(4)	*(paper, conf)*	*~ (pub, //name)*
(5)	*(conf, .conf)*	*~ (conf, name)*
(6)	*(author, name)*	*~ (writer, writer)*

Table 6. Compatibility of terminal edges

(M1)	*//paper*[. = x_2] ⇔ *//pub*[*title*[. = x_2]]
(M2)	*//paper*[*year*[. = x_3]] ⇔ *//pub*[*//year*[. = x_3]]
(M3)	*//conf*[*../year* = x_3] ⇔ *//conf*[*year* = x_3]
(M4)	*//paper[conf[. = x_4]]* ⇔ *//pub[//name[. = x_4]]*
(M5)	*//conf*[. = x_4] ⇔ *//conf*[*name*[. = x_4]]
(M6)	*//author*[*name*[. = x_1]] ⇔ *//writer*[. = x_1]

Table 7. Matching formulas for terminal edges (from Table 6)

- $M9 = R4(E2 \sim E8^{-1}, M6, E8 \sim E2^{-1}, M7)$,
- $M10' = R5(M4, M8, E4 \sim E6)$,
- $M10 = minimize(M10')$,
- $M12 = R2(E1 \sim E5, M11)$.

Using (M9), and after removing ambiguities (resulting from occurrences of //), the following mapping between subschemas of D_1 ans D_2 can be obtained (see also (2)):

//*author*[*name*[. = x_1], *paper*[. = x_2, *year*[. = x_3],
conf[. = x_4]]] ⇒ /*result*[*pub*[*title* . = x_2],
writer[. = x_1], *conf*[*year*[. = x_3], *name*[. = x_4]]]

Note that mapping specifies how data from the source instance is to be transformed into the structure described by the right-hand side (the target structure). It means, that the right-hand side must unambiguously determine the target structure, thus the operator // may not occur in the right-hand side TPF.

6. Conclusion

In this paper, we propose a method for discovering XML schema matching and mapping using semantic annotation of XML schemas, where a schema is understood as a graph over the underlying DTD. The graph consists of labels and generalized edges determined by the DTD. The predecessor of an edge is a label and the successor – a label reachable from the predecessor by means of a simple operator (self, child, descendent or sibling). Labels and edges are annotated in a common domain ontology. The annotation forms the base for deriving matches and mappings between schemas annotated in the same ontology. We propose a set of rules to infer these relationships.

(M7)	//*paper*[. = x_2, *year*[. = x_3], *conf*[. = x_4]] ⇔ //*pub*[*title*[. = x_2], //*year*[. = x_3], //*name*[. = x_4]]
(M8)	//*conf*[../*year*[. = x_3],. = x_4] ⇔ //*conf*[*year*[. = x_3], *name*[. = x_4]]
(M9)	//*author*[*name*[. = x1], *paper*[. = x_2, *year*[. = x_3], *conf*[. = x_4]]] ⇔ //*pub*[*title*[. = x_2], //*year*[. = x_3], //*name*[. = x_4], *writer*[. = x1]]
(M10´)	//*paper*[*conf*[. = x_4, ../*year*[. = x_3],. = x4]] ⇔ //*pub*[//*name*[. = x_4], *conf*[*year*[. = x_3], *name*[. = x_4]]]
(M10)	//*paper*[*year*[. = x_3], *conf*[. = x_4]] ⇔ //*pub*[*conf*[*year*[. = x_3], *name*[. = x_4]]]
(M11)	//*author*[*name*[. = x_1], *paper*[. = x_2, *year*[. = x_3], *conf*[. = x_4]]] ⇔ //*pub*[*title*[. = x_2], *conf*[*year*[. = x_3], *name*[. = x_4]], *writer*[. = x1]]
(M12)	//*authors*[*author*[*name*[. = x_1], *paper*[. = x_2, *year*[. = x_3], *conf*[. = x_4]]]] ⇔ //*pubs*[*pub*[*title*[. = x_2], *conf*[*year*[. = x_3], *name*[. = x_4]], *writer*[. = x_1]]

Table 8. Complex matching formulas

The proposed method of XML schema annotation and inferring schema mappings is implemented in SixP2P (*Semantic integration of XML data in P2P*) system [15]. In SixP2P implementation, schema mappings are translated into XQuery programs [13].

Acknowledgement

The work was supported in part by the Polish Ministry of Science and Higher Education under grant 3695/B/T02/2009/36, and in part under grant 45083/DS/10.

References

[1] Abiteboul, S., R., Hull, Vianu, V. (1995) . *Foundations of Databases*. Addison-Wesley, Reading, Massachusetts.

[2] Amano, S., Libkin, L.,Murlak, F (2009). XML Schema Mappings. *In*: PODS Conference, p. 33–42.

[3] Arenas, M., Libkin, L (2005). XML Data Exchange: Consistency and Query Answering. *In*: PODS Conference, p. 13–24.

[4] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Petel-Schneider, P. ed. (2003). The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press.
[5] Beneventano., D., Bergamaschi, S. (2004). The MOMIS methodology for integrating heterogeneous data sources. *In*: IFIPCongress Topical Sessions, p. 19–24.
[6] Bex, G. J., Neven, F., den Bussche, J. V (2004). Dtds versus xml schema: A practical study. *In* WebDB, p. 79–84.
[7] Euzenat, J. (2007). Semantic precision and recall for ontology alignment evaluation. *In*: M. M. Veloso, ed. *IJCAI*, p. 348– 353.
[8] Fagin, R., Haas, L. M., Hernández, M. A., Miller, R. J., Popa, L., Velegrakis, Y. (2009). Clio: Schema mapping creation and data exchange. *In*: *Conceptual Modeling: Foundations and Applications*, volume Lecture Notes in Computer Science 5600, p.198–236.
[9] Fagin, R., Kolaitis, P. G., Popa, L., Tan, W. C. (2004). Composing schema mappings: Second-order dependencies to the rescue. *In*: PODS, p. 83–94.
[10] Madhavan, J., Bernstein, P. A., Doan, A., Halevy, A. Y. (2005). Corpus-based schema matching. *In*: Proceedings of the 21stInternational Conference on Data Engineering, ICDE, p. 57–68. IEEE Computer Society.
[11] Martens, W., Neven, F., Schwentick, T (2007). Simple off the shelf abstractions for XML schema, *SIGMOD Record*, 36 (3) 15–22.
[12] OWL (2004). Web Ontology Language Overview. w3.org/TR/owlref
[13] Pankowski, T. (2008). XML data integration in SixP2P – a theoretical framework. *In*: EDBT Workshop Data Management in P2P Systems (DAMAP 2008), ACM Digital Library, p. 11–18.
[14] Pankowski, T. (2008). XML Schema Mappings Using Schema Constraints and Skolem Functions. *In*: Knowledge Engineering and Intelligent Computations, p. 199–216. Knowledge-Driven Computing, Springer Verlag.
[15] Pankowski, T. (2009). Schema mappings and annotations in semantic integration of XML data in P2P data integration systems. *In*: Application of Digital Information and Web Technologies (ICADIWT´09), p. 505–510.
[16] Pankowski, T., Cybulka, J., Meissner, A (2007). XML Schema Mappings in the Presence of Key Constraints and Value Dependencies. *In*: ICDT 2007 Workshop EROW´07, p. –15. CEUR Workshop Proceedings, CEURWS. org,V. 229.
[17] Pankowski, T., Hunt,. E (2005). Data merging in life science data integration systems. *In*: Intelligent Information Systems, New Trends in Intelligent Information Processing and Web Mining, p. 279–288. Advances in Soft Computing, Springer Verlag.
[18] Rahm, E., Bernstein,. P. A (2001). A survey of approaches to automatic schema matching. The VLDB Journal, 10 (4) 334–350.
[19] Shvaiko, P., Euzenat, J. (2005). A survey of schema-based matching approaches. *J. Data Semantics IV*, 3730. 146–171.
[20] Xiao, H., Cruz, I. F. (2006). Integrating and Exchanging XML Data Using Ontologies. *Journal on Data Semantics VI:Special Issue on Emergent Semantics,* Lecture Notes in Computer Science 4090*,* Springer, p. 67–89.
[21] XML Path Language (XPath) 2.0, 2006. www.w3.org/TR/xpath20.
[22] Yu, C., Popa, L. (2004). Constraint-Based XML Query Rewriting For Data Integration. *In: SIGMOD Conference*, p. 371– 382.