Detecting semantics-preserving XML schema mappings based on annotations to OWL ontology

Tadeusz Pankowski

Poznań University of Technology, Poland

EDBT/ICDT Workshop on Logic in Databases, 2011
Outline

Motivation
   Semantics preservation in schema mappings & in DE
   Kinds of schema mappings

XML schemas and OWL ontology
   Tree patterns and tree-pattern formulas
   Representation of OWL ontology

Defining semantics for XML data
   Schema annotation
   XML-to-OWL mappings

Reasoning about tree patterns
   Example

Semantics-preserving XML-to-XML mappings

Conclusions
Schema mappings

A schema mapping is a triple [Fagin:2005]

\[ M = (S, T, \Sigma), \]

- \( S \) – a source schema,
- \( T \) – a target schema,
- \( \Sigma \) – a set of source-to-target dependencies (STDs), i.e. first-order sentences of the form

\[ \forall u(\varphi(u) \Rightarrow \exists w \psi(u, w)), \]

where \( \varphi(u) \) and \( \psi(u, w) \) are conjunction of atomic formulas over \( S \) and \( T \); \( u \) and \( w \) are vectors of variables.

If \( I_1 \models S, I_2 \models T \), and \( (I_1, I_2) \models \Sigma \) then \( I_2 \) is a solution of \( I_1 \) w.r.t. \( M \), \( I_2 \in \text{Sol}_M(I_1) \).

(\( M \) specifies how instances of \( S \) are to be transformed into instances of \( T \))
Semantics preservation in schema mappings

Let

\[ I_2 \in Sol_M(I_1) \]

What about semantics of data? The rule

\[ \text{paper} \rightarrow \text{author}+ \]

might be interpreted as the property

\[ \text{paper WrittenBy author} \]

in schema \( S_1 \), and as the property

\[ \text{paper CitedBy author} \]

in schema \( S_2 \). Then data exchange between \( S_1 \) and \( S_2 \) is not semantics-preserving.
Kinds of schema mappings

\[ M = (S, T, \Sigma), \]
\[ \sigma = \forall u (\varphi(u) \Rightarrow \exists w \psi(u, w)). \]

A schema defines a set of WFF, admissible atomic formulas, for the source, \( \varphi(u) \), and target, \( \psi(u, w) \), formulas in STD. We will consider:

- **OWL ontology schema** (a relational schema), \( O \),
- **XML schemas** (DTD-oriented), \( S_1, S_2, \ldots \),

and four kinds of schema mappings between them:

- XML-to-OWL mappings,
- OWL-to-OWL mappings,
- OWL-to-XML mappings,
- XML-to-XML mappings.
XML schemas and tree patterns

An XML schema is a tuple \( S = (\text{top}, \text{Lab}, \#\text{PCDATA}, \rho) \),
where: \( \rho : l \rightarrow e, l \in \text{Lab}, e \) – a regular expression

\[
e ::= \#\text{PCDATA} \mid l \mid l? \mid l^+ \mid l^* \mid e e.
\]

\( \text{bib} \rightarrow \text{paper*}, \text{paper} \rightarrow \text{title} \text{ type},\)
\( \text{title} \rightarrow \#\text{PCDATA}, \text{type} \rightarrow \#\text{PCDATA} \)

Tree patterns over XML schema (examples)

\[
\pi_1 = \text{paper?}[\text{title[]}, \text{type["c"]}],
\pi_2 = \text{cPaper?}[\text{title[]}]
\]

Syntax:

\[
\pi ::= \varepsilon \mid c \mid l \mid l[\pi, \ldots, \pi] \mid l? \mid l?[\pi, \ldots, \pi].
\]

Elementary tree patterns:

\(l, l[l'], l[]\).
XML trees and tree-pattern formulas

- An XML tree is a tuple $I = (r, N, Child, \lambda, \nu)$; $I \models S$, $I \models \pi$.

- Tree formulas over tree patterns (example):

  $\pi_1(x_1, x_2, x_3, v_1) = \text{paper}?(x_1)[\text{title}(x_2)[v_1], \text{type}(x_3)["c"]]
  \pi_2(x_4, x_5, v_2) = \text{cPaper}?(x_4)[\text{title}(x_5)[v_2]]

- Tree-tuple formulas over tree patterns (example):

  $\pi_1(v_1) = \text{paper}?[\text{title}[v_1], \text{type}["c"]]
  \pi_2(v_2) = \text{cPaper}?[\text{title}[v_2]]
A (part of) OWL ontology, $O$, is represented as follows:

1. **Ontology schema** – three relation symbols (binary and ternary)

   $$O = (CA, OPA, DPA),$$

   - $CA$ – ClassAssertions, $CA(C, y)$,
   - $OPA$ – ObjectPropertyAssertions, $OPA(R, y_1, y_2)$,
   - $DPA$ – DataPropertyAssertions, $DPA(D, y, v), DPA(D, y, c)$.

2. A set $\Delta$ of **axioms**, i.e. a set of OWL-to-OWL STDs of the form

   $$\forall y_1, v(\psi_1(y_1, v) \Rightarrow \exists y_2 \psi_2(y_1, y_2, v)),$$

   $y_1, y_2$ – vectors of variables valuated with ontology’s individuals, $v$ – a vector of variables valuated with text values (literals).

3. A set $\mathcal{O}$ of sentences built over $\mathcal{O}$ and satisfying $\Delta$, is an **instances** of $\mathcal{O}$. 
Example of OWL ontology

Terminology:

\[\text{Class} = \{\text{Bibliography}, \text{Paper}, \text{Title}, \text{Type}, \text{ConfPaper}, \text{JournalPaper}, \ldots\}\],

\[\text{ObjectProperty} = \{\text{Contains}, \text{hasTitle}, \text{hasType}, \ldots\}\],

\[\text{DataProperty} = \{\text{valueOfTitle}, \text{valueOfType}, \ldots\}\].

Axioms, \(\Delta:\)

\[(A1)\] \(\text{CA}(\text{ConfPaper}, y) \Rightarrow \text{CA}(\text{Paper}, y)\)

\[
\ldots \ldots
\]

\[(A5)\] \(\text{OPA}(\text{hasType}, y_1, y_2) \Rightarrow \text{CA}(\text{Paper}, y_1) \land \text{CA}(\text{Type}, y_2)\)

\[(A6)\] \(\text{CA}(\text{Paper}, y_1) \land \text{OPA}(\text{hasType}, y_1, y_2)\)

\[
\land \text{DPA}(\text{valueOfType}, y_2, "c") \Rightarrow \text{CA}(\text{ConfPaper}, y_1)
\]

\[(A7)\] \(\text{CA}(\text{ConfPaper}, y_1) \Rightarrow \exists y_2 \ (\text{OPA}(\text{hasType}, y_1, y_2)\)

\[
\land \text{DPA}(\text{valueOfType}, y_2, "c")\))\)
1. **Annotation** $(\alpha)$ of elementary tree patterns:
   - $l \mapsto C, C \in \text{Class};$
   - $l[l'] \mapsto R, R \in \text{ObjectProperty};$
   - $l[] \mapsto D, D \in \text{DataProperty}.$

2. **Example** Annotation of $S_1$ into $O$:
   - $\text{bib} \mapsto \text{Bibliography}, \text{paper} \mapsto \text{Paper}, \text{title} \mapsto \text{Title}, \text{type} \mapsto \text{Type};$
   - $\text{bib}[\text{paper}] \mapsto \text{Contains}, \text{paper}[\text{title}] \mapsto \text{hasTitle}, \text{paper}[\text{type}] \mapsto \text{hasType};$
   - $\text{title[]} \mapsto \text{valueOfTitle}, \text{type[]} \mapsto \text{valueOfType}.$
Algorithm 1 – Generating dependent OWL formula

Input: A tree formula $\pi(x, v)$ over a schema $S$ and an annotation $\alpha$ of $S$ into OWL ontology $O$.

Output: $\psi(x, v) = \tau_\alpha(\pi(x, v))$ – an OWL formula dependent on $\pi(x, v)$ and $\alpha$.

1. if $l \mapsto C$, and $l[] \mapsto D$ then

   $\tau_\alpha(l(x)[c]) := CA(C, h(x)) \land DPA(D, h(x), c),$ 
   $\tau_\alpha(l(x)[v]) := CA(C, h(x)) \land DPA(D, h(x), v).$

2. if $l \mapsto C$, and $l[l'] \mapsto R$ then

   $\tau_\alpha(l(x)[l'(x')[\pi(x, v)]])) :=$ 
   $CA(C, h(x)) \land OPA(R, h(x), h(x')) \land \tau_\alpha(l'(x')[\pi(x, v)]).$

3. $\tau_\alpha(l(x)[\pi_1(x_1, v_1), \ldots, \pi_k(x_k, v_k)]) :=$ 
   $\tau_\alpha(l(x)[\pi_1(x_1, v_1)]) \land \cdots \land \tau_\alpha(l(x)[\pi_k(x_1, v_k)]).$
1. XML-to-OWL dependency (full)

\[ \forall x, v(\pi(x, v) \to \psi(h(x), v)). \]

2. Example

\[ \forall x_1, x_2, x_3, v(\pi_1(x_1, x_2, x_3, v) \Rightarrow \psi_1(h(x_1), h(x_2), h(x_3), v)), \]

\[ \pi_1(x_1, x_2, x_3, v) = \text{paper?}(x_1)[\text{title}(x_2)[v], \text{type}(x_3)["c" ]] \]

\[ \psi_1(h(x_1), h(x_2), h(x_3), v) = \text{CA}(\text{Paper}, h(x_1)) \land \text{CA}(\text{Title}, h(x_2)) \land \text{CA}(\text{Type}, h(x_3)) \land \text{OPA}(\text{hasTitle}, h(x_1), h(x_2)) \land \text{OPA}(\text{hasType}, h(x_1), h(x_3)) \land \text{DPA}(\text{valueOfTitle}, h(x_2), v) \land \text{DPA}(\text{valueOfType}, h(x_3), "c") \]
Semantic dependencies between XML data and OWL ontology

OWL ontology:

XML tree:

$h : \text{Node} \rightarrow \text{Individual}$.

Preimage – a set of semantically equivalent nodes:

$$h^{-1}(y) = \{ x | h(x) = y \}.$$
Reasoning about tree patterns

\[ \mu_1 = (\pi_1, O, \{ \forall x_1, v_1 \pi_1(x_1, v_1) \Rightarrow \psi_1(h(x_1), v_1) \}) \]
\[ \mu_2 = (\pi_2, O, \{ \forall x_2, v_2 \pi_2(x_2, v_2) \Rightarrow \psi_2(h(x_2), v_2) \}) \]

\[ \pi_1 \sqsubseteq \pi_2, \text{ if by a "consistent substitution of variables for terms" unification} \]

\[ \psi_1(y, v) \Rightarrow \exists y' \psi_2(y, y', v) \]
Algorithm 2 – Creating deduction formula (unification)

Input: Two XML-to-OWL mappings, $\mu_1$ and $\mu_2$, from XML schemas to OWL ontology $O$, and the set $\Delta$ of axioms in $O$.

Output: A deduction formula of the form (1).

1. Proceeding with $\psi_1$:
   - Each occurrence of a term $t$ (i.e. $h(x_i)$, $y$, or $v$) in $\psi_1$ replace with such a variable that does not occur in any axiom in $\Delta$.

2. For each axiom $\delta$ in $\Delta$:
   - If $\text{CA}(C,y)$ is in $\Delta$ and $\text{CA}(C,y')$ is in $\psi_1$, then replace each occurrence of $y$ in $\delta$ with $y'$.
   - If $\text{OPA}(R,y_1,y_2)$ is in $\Delta$ and $\text{OPA}(R,y'_1,y'_2)$ is in $\psi_1$, then replace each occurrence of $y_1$ in $\delta$ with $y'_1$ and each occurrence of $y_2$ in $\delta$ with $y'_2$.
   - If $\text{DPA}(D,y,v)$ is in $\Delta$ and $\text{DPA}(D,y',v')$ is in $\psi_1$, then replace each occurrence of $y$ in $\delta$ with $y'$ and each occurrence of $v$ in $\delta$ with $v'$.

3. Proceeding with terms in $\psi_2$:
   - Each occurrence of a term $t$ (i.e. $h(x_i)$, $y$, or $v$) in $\psi_2$ replace with such a variable that does not occur neither in any axiom in $\Delta$, nor in $\psi_1$.
   - If $\text{CA}(C,y)$ is in $\psi_2$, and $\text{CA}(C,y')$ is in $\delta$ or in $\psi_1$, then replace each occurrence of $y$ in $\psi_2$ with $y'$. Similarly, for variables $y_1, y_2, y$ occurring in $\text{OPA}(R,y_1,y_2)$ and $\text{DPA}(D,y,v)$.
   - The set $y'$ of variables not occurring in $\psi_1$ constitutes the sets of existentially quantified variables.

4. Construct the final expected deduction rule (1).
Reasoning about tree patterns – example

Example

cPaper[title[]] ⊆ paper[title[], type["c"]].

Consistent substitution of variables in axioms (A1), (A5) and (A7):

\[
\frac{A1(p), A5(p, y_2), A7(p, y_2)}{\psi_2(p, t, v) ⇒ \exists y_2 \psi_1(p, t, y_2, v)}.
\]

\[
CA(ConfPaper, p) ⇒ CA(Paper, p),
\]

\[
OPA(hasType, p, y_2) ⇒ CA(Paper, p) ∧ CA(Type, y_2),
\]

\[
CA(ConfPaper, p) ⇒ \exists y_2 (OPA(hasType, p, y_2) ∧ DPA(valueOfType, y_2, ”c”))
\]

\[
CA(ConfPaper, p) ∧ CA(Title, t) ∧ OPA(hasTitle, p, t) ∧ DPA(valueOfType, t, v) ⇒
\]

\[
\exists y_2 (CA(Paper, p) ∧ CA(Title, t) ∧ CA(Type, y_2) ∧ OPA(hasTitle, p, t) ∧ OPA(hasType, p, y_2) ∧ DPA(valueOfType, t, v) ∧ DPA(valueOfType, y_2, ”c”)).
\]
XML-to-XML mapping – second order

Theorem

Let \( \pi_1 \sqsubseteq \pi_2 \), then there exists such an XML-to-XML mapping \( \mathcal{M} = (\pi_1, \pi_2, \{\sigma\}) \), where

\[
\sigma = \exists f_1, \ldots, \exists f_k \, \forall x_1, v \ (\pi_1(x_1, v) \Rightarrow \exists x_2 \in h^{-1}(h(x_1), h(f_1(x_1)), \ldots, h(f_k(x_1))) \pi_2(x_2, v)),
\]

that for each instance \( I_1 \) of \( \pi_1 \) there exists such an instance \( I_2 \) of \( \pi_2 \) that \( (I_1, I_2) \models \mathcal{M} \).
Semantics-preserving XML-to-XML mappings

\( \mathcal{M} \) is a semantics-preserving mapping from \( \pi_1 \) to \( \pi_2 \), iff

\[
\exists \mu_1, \mu_2, m \ \forall I_1, I_2, O_1, O_2 \ (I_1, O_1) \models \mu_1 \land (I_2, O_2) \models \mu_2 \land (O_1, O_2) \models m \land \Delta \vdash m \\
\Rightarrow \mathcal{M} = \mu_1 \circ m \circ \mu_2^{-1}.
\]

Then \( I_2 \) is a semantics-preserving solution of \( I_1 \) w.r.t. \( \mathcal{M} \), i.e.

\[
I_2 \in SemPresSol_\mathcal{M}(I_1).
\]
XML-to-XML mappings using tree-tuple formulas

Example

For our running example, we have

$$\exists f \forall x_1, x_2, v \ (cPaper(x_1)[x_2][v])$$

$$\Rightarrow \exists (x'_1, x'_2, x'_3) \in h^{-1}(h(x_1), h(x_2), h(f(x_1, x_2)))$$

$$\quad paper(x'_1)[title(x'_2)[v], type["c"]])$$

The mapping can be restricted to tree-tuple formulas:

$$\forall v (cPaper[title[v]] \Rightarrow paper[title[v], type["c"]).$$
SixP2P – Semantic Integration of XML data in P2P environment
Conclusions

1. Semantics-preserving schema mappings have been defined and discussed.

2. Semantics of data is defined by means of annotation into OWL ontology.

3. Reasoning within OWL ontology is used to establish semantic subsumption between tree patterns (schemas).

4. The method underlines implementation of SixP2P system (Semantic integration of XML data in P2P environment).
Thank You!