On Inference of XML Schema with the Knowledge of an Obsolete One

Irena Mlýnková

Department of Software Engineering
Faculty of Mathematics and Physics
Charles University
Prague, Czech Republic

mlynkova@ksi.mff.cuni.cz
http://www.ksi.mff.cuni.cz/~mlynkova/
Overview

1. Introduction
2. Existing Approaches
3. Proposed Improvement
4. Conclusion
Introduction

- XML = a standard for data representation and manipulation
- XML documents + XML schema
  - Allowed data structure
  - W3C recommendations: DTD, XML Schema (XSD)
  - ISO standards: RELAX NG, Schematron, ...
- Why schema?
  - Known structure, valid data, limited complexity
    ⇒ Optimization
    - Storing, querying, updating, compressing, ...

January 19 - 23, 2009

ACSW'09 - Wellington, New Zealand
Real-World XML Schemas

- Statistical analyses of real-word XML data:
  - 52% of randomly crawled / 7.4% of semi-automatically collected documents: no schema
  - 0.09% of randomly crawled / 38% of semi-automatically collected documents with schema: use XSD
  - 85% of randomly crawled XSDs: equivalent to DTDs

- Problem:
  - Users do not use schemas at all
  - Schema = a kind of documentation
    - Documents are not valid, schemas are not correct
  - XML Schema language is not used
    - Too complex, too difficult
Inference of XML Schemas

Solution:
- Automatic inference of XML schema $S_D$ for a given set of documents $D$
  - Multiple solutions
    - Too general = accepts too many documents
    - Too restrictive = accepts only $D$

Advantages:
- $S_D$ = a good initial draft for user-specified schema
- $S_D$ = a reasonable representative when no schema is available
- User-defined XML schemas are too general (*, +, recursion, ...) $\Rightarrow S_D$ can be more precise
An extended context-free grammar is quadruple $G = (N, T, P, S)$, where $N$ and $T$ are finite sets of nonterminals and terminals, $P$ is a finite set of productions and $S$ is a non terminal called a start symbol. Each production is of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha$ is a regular expression over alphabet $N \cup T$.

Given the alphabet $\Sigma$, a regular expression (RE) over $\Sigma$ is inductively defined as follows:
- $\emptyset$ (empty set) and $\varepsilon$ (empty string) are REs
- $\forall a \in \Sigma : a$ is a RE
- If $r$ and $s$ are REs over $\Sigma$, then $(rs)$ (concatenation), $(r|s)$ (alternation) and $(r^*)$ (Kleene closure) are REs

- DTD adds: $(s|\varepsilon) = (s?)$, $(s \ s^*) = (s+)$, concatenation = ','
- XML Schema adds: unordered sequence
Overview

1. Introduction
2. Existing Approaches
3. Proposed Improvement
4. Conclusion
Existing Approaches

- Main focus: Inference of REs (content models)
  - DTD aspect
  - Aim: Concise and precise
- Gold's theorem: Regular languages are not identifiable in the limit only from positive examples (valid XML documents)
  - Heuristic = no theoretic basis
    - Generalization of a trivial schema
    - Rules: “If there are > 3 occurrences of element E, it can occur arbitrary times ⇒ E+ or E*”
- Inferring a grammar
  - Inference of identifiable subclasses of regular languages
Classical Steps

1. Derivation of initial grammar (IG)
   - For each element E and its subelements E₁, E₂, ..., Eₙ we create
     production \( E \rightarrow E₁ E₂ \ldots Eₙ \)

2. Clustering of rules of IG
   - According to element names vs. broader context

3. Construction of prefix tree automaton (PTA) for each cluster

4. Generalization of PTAs
   - Merging state algorithms

5. Inference of simple data types and integrity constraints
   - Often ignored

6. Refactorization
   - Correction and simplification of the derived REs

7. Expressing the inferred REs in target XML schema language
   - Most common: Direct rewriting of REs to content models
Example (1)

...<person id="123">
  <name>
    <first>Irena</first>
    <surname>Mlynkova</surname>
  </name>
  <email>irena.mlynkova@gmail.com</email>
  <email>irena.mlynkova@mff.cuni.cz</email>
</person>

<person id="456" holiday="yes">
  <name>
    <surname>Necasky</surname>
    <first>Martin</first>
  </name>
  <phone>123-456-789</phone>
  <email>martin.necasky@mff.cuni.cz</email>
</person>
...

person → name email email
person → name phone email
name → first surname
name → surname first
first → PCDATA
surname → PCDATA
email → PCDATA
phone → PCDATA

person:

1 --- 2 --- 4 --- 5

phone

6 --- 7

email
Example (2)

person → name email address
person → name address

person → name email address
person → name phone address

1 → 2 → 3 → 4
name email address

address

5

1 → 2 → 3 → 4
name email address

phone

5 → 6
address

1 → 2 → 3 → 4
name email address

phone

1 → 2 → 35 → 46
name

email

address

person → name email? address

person → name (email | phone) address
Overview

1. Introduction
2. Existing Approaches
3. Proposed Improvement
4. Conclusion
Our Approach

- Assumption: We are provided with the original, but already obsolete schema $S_{\text{orig}}$
  - Analysis of real-world XML data: quite common situation
    - Schema = a kind of documentation
    - Schema is not used for validation of XML data
      $\Rightarrow$ not updated with the data

- Idea: Exploitation of the information which was correct once

$\Rightarrow$ Aim:
  - Optimization of the inference approach
  - Exploitation of useful source of information
General Observations

- General idea: Checking correctness and adaptation of
  - Simple data types
    - Trivial ⇒ generally ignored
  - Element/attribute names
    - Equivalent cases vs. semantic similarity ⇒ finding a mapping
  - REs

- Possible situations:
  1. Documents in D are valid against $S_{orig}$; $S_{orig}$ is enough concise and precise
  2. Documents in D are valid against $S_{orig}$; $S_{orig}$ is too general
  3. Documents in D are not valid against $S_{orig}$
Proposed Solution

- **Step 1: Correction of the input schema**
  - **Assumption:** \( \exists d \in D \text{ s.t. } d \text{ is not valid against } S_{\text{orig}} \)
  - **Aim:** to find schema \( S_{\text{correct}} = \text{correction of } S_{\text{orig}} \) s.t.
    - For \( \forall d \in D : d \text{ is valid against } S_{\text{correct}} \)
    - \( \text{dist}(S_{\text{orig}}, S_{\text{correct}}) \leq \text{dist}(S_{\text{orig}}, S) \) for \( \forall S \in \Sigma_{\text{correct}} \), where \( \Sigma_{\text{correct}} \) is the set of all possible corrections of \( S_{\text{orig}} \)
  - \( \Rightarrow \) Output: \( S_{\text{correct}} \)

- **Step 2: Specialization of the input schema**
  - **Assumption:** \( \forall d \in D : d \text{ is valid against } S_{\text{correct}} \)
  - **Aim:** to specialize REs in \( S_{\text{correct}} \) with regard to \( D \)
  - \( \Rightarrow \) Output: \( S'_{\text{correct}} \)
Step 1. Schema Correction

- Schema correction:
  1. We divide $D$ into sets $D_{\text{valid}}$ and $D_{\text{invalid}}$ s.t. $D_{\text{valid}} \cup D_{\text{invalid}} = D$ and $D_{\text{valid}} \cap D_{\text{invalid}} = \emptyset$
  2. For $\forall \in D_{\text{invalid}}$ we create the respective set of productions $\{p_1, p_2, \ldots, p_m\}$ and merge them with $S_{\text{orig}} = \{q_1, q_2, \ldots, q_n\}$

- Merging strategy for $p_i$
  1. Finding $q_j$ to be merged with
    - Same as the original clustering strategy
  2. Parsing of model($p_i$) and checking validity against model($q_j$) $\Rightarrow$ PTA
  3. Merging states of PTA
    - Modified strategy
Example

$q_j$:

$p_i$:

PTA:

Merged:
Merging State Strategy

- Combinatorial optimization problem (COP)
  - A search space $\Sigma$ of solutions (feasible region)
  - A set $\Omega$ of constraints over $\Sigma$
  - Evaluation function $f : \Sigma \rightarrow R_0^+$ (objective function)

- Our case:
  - $\Sigma = \text{a set of possible generalizations of input schema } S_{\text{input}}$
  - $\Omega$ is given by the features of XML schema language
  - $f = \text{evaluates the quality of given } S \in \Sigma$

- MDL (Minimum Description Length) principle
  - Good schema is enough general $\Rightarrow$ low number of states of automaton
  - Good schema preserves details $\Rightarrow$ express instances using short codes

- Problem: $\Sigma$ is theoretically infinite $\Rightarrow$ heuristics $\Rightarrow$ suboptimal solution
  - Search algorithm: ACO (Ant Colony Optimization)
Ant Colony Optimization (ACO)

- Meta-heuristics for solving COPs
- Idea: Artificial ants iteratively search space $\Sigma$ and improve $S_{\text{input}}$
- Ant
  - Searches a subspace of $\Sigma$ until it “dies”
    - After performing $N_{\text{ant}}$ steps
  - Spreads “pheromone”
    - Positive feedback = how good solution it has found so far
  - Exploits spread pheromone of other ants to select next step
    - Step = a possible way of schema generalization
    - Selected randomly, probability is given by $f$
Possible Steps of Ants

- Existing works:
  - k,h-context method: “Two states $t_x$ and $t_y$ of an automaton are identical if there exist identical paths of length $k$ terminating in $t_x$ and $t_y$.”
  - s,k-string method: Nerod’s equivalency: “Two states $t_x$ and $t_y$ of an automaton are equivalent if all paths of length $k$ leading from $t_x$ and $t_y$ are equivalent.”

- Problem: We do not want to modify the original automaton
- Solution: We merge only if the set of merged states involves at least one of the states of the new branch

- Situations:
  1. We truncate the new branch
     - Merging within the branch
  2. We reduce the number of states of the whole automaton
     - Merging of new states with original ones
Step 2. Schema Specialization

- Assumption: \( \forall d \in D : d \text{ is valid against } S_{\text{correct}} \)
- Aim: to specialize REs involved in \( S_{\text{correct}} \) with regard to \( D \)
  \[ \Rightarrow \] Output: \( S'_{\text{correct}} \)
- Idea: Parsing of documents in \( D \) and checking preciseness of \( S'_{\text{correct}} \)
- Steps:
  1. Pruning of unused schema fragments
  2. Correction of lower and upper bounds of occurrences of schema fragments
  3. Correction of operators
  4. Refactorization
1. Pruning of Unused Schema Fragments

- Idea: $\forall e \in S_{\text{correct}}$ we set usage flag $\varphi_{\text{used}}(e)$

- Example:

  - $E \to A \ C$
  - $E \to A \ B \ B \ B \ C$
  - $E \to A \ B \ C$
  - $E \to A \ B \ B \ B \ B$

- Note: Elimination of unused schema fragments preserves correctness of content models
2. Correction of Lower and Upper Bounds

- Idea: \( \forall e \in S_{\text{correct}} \) we set minimum and maximum repetition flag \( \phi_{\text{min}}(e) \) and \( \phi_{\text{max}}(e) \)

- Note: \( \phi_{\text{min}}(e) \) and \( \phi_{\text{max}}(e) \) cover \( \phi_{\text{used}}(e) \)
3. Correction of Operators

4. Factorization

- **Restriction of content models**
  - Can be applied only if validity is not violated

- **Improving readability, simplification of structure**
  - Classical step

<table>
<thead>
<tr>
<th>Restriction of content models</th>
<th>Restriction of content models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be applied only if validity is not violated</td>
<td>Math operations to simplify structure</td>
</tr>
<tr>
<td>Improving readability, simplification of structure</td>
<td>Classical step</td>
</tr>
</tbody>
</table>

- Classical step

<table>
<thead>
<tr>
<th>Restriction of content models</th>
<th>Restriction of content models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be applied only if validity is not violated</td>
<td>Math operations to simplify structure</td>
</tr>
<tr>
<td>Improving readability, simplification of structure</td>
<td>Classical step</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a?? → a?</th>
<th>a?? → a?</th>
</tr>
</thead>
<tbody>
<tr>
<td>a++ → a+</td>
<td>a++ → a+</td>
</tr>
<tr>
<td>a** → a*</td>
<td>a** → a*</td>
</tr>
<tr>
<td>a*? → a*</td>
<td>a*? → a*</td>
</tr>
<tr>
<td>a?* → a*</td>
<td>a?* → a*</td>
</tr>
<tr>
<td>a++ → a*</td>
<td>a++ → a*</td>
</tr>
<tr>
<td>a*+ → a*</td>
<td>a*+ → a*</td>
</tr>
<tr>
<td>a?+ → a*</td>
<td>a?+ → a*</td>
</tr>
<tr>
<td>a?+ → a*</td>
<td>a?+ → a*</td>
</tr>
<tr>
<td>(ab)(ac) → a(b</td>
<td>c)</td>
</tr>
</tbody>
</table>
Complexity of Algorithm

- Schema correction:
  - ACO heuristic
  - Worst case:
    - Allowed number of iterations, number of steps of an ant, number of ants

- Schema specialization:
  - Linear parsing of documents in D and content models in $S_{\text{correct}}$
  - In the worst case: $S_{\text{orig}}$ provides no useful information $\Rightarrow$ same complexity as in the original algorithm
  - Checking of correctness is linear

- Otherwise: We start with partly inferred schema
Overview

1. Introduction
2. Existing Approaches
3. Proposed Improvement
4. Conclusion
Conclusion

- Advantages of algorithm:
  - Optimization of the inference process
  - Exploitation of available source of information
  - Exploitation of verified approaches
    - ACO, MDL, merging state algorithm, ...

- Current and future work
  - Implementation
    - Other improvements ⇒ mutual comparison of impact
  - Exploitation
    - Storage strategies of XML data
  - Further improvements
    - User interaction, inference of integrity constraints, other schema languages (RELAX NG, Schematron)...

January 19 - 23, 2009  ACSW'09 – Wellington, New Zealand
Thank you