Interactive Inference of XML Schemas

Julie Vyhnanovska
Irena Mlynkova

julie.vyhnanovska@gmail.com
irena.mlynkova@mff.cuni.cz

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

- XML = a standard for data representation and manipulation
  - XML documents + XML schema
    - DTD, XML Schema, Schematron, RELAX NG, ...
- Why schema?
  - Known structure, valid data, ...
  - Limited complexity ⇒ optimization
- Problems of real-world data:
  - Users do not use schemas at all
    - Schema = a kind of documentation
  - XML Schema (W3C) language is not used
- Solution: Automatic inference of XML schema $S_D$ for a given set of documents $D$

Existing Approaches

- Fact: XML schema = extended context-free grammar

- Classical steps:
  1. Derivation of initial grammar (IG)
     - For each element $E$ and its subelements $E_1, E_2, \ldots, E_n$ we create production $E \rightarrow E_1 E_2 \ldots E_n$
  2. Clustering of rules of IG
  3. Construction of prefix tree automaton (PTA) for each cluster
  4. Generalization of PTAs
     - Merging state algorithms
     - Multiple solutions:
       - We need to evaluate the quality of a solution
       - Too general vs. too restrictive
  5. 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  
ext → PCDATA  
phone → PCDATA  

person:  

1 — name —> 2 — email —> 4 — email —> 5 — phone —> 6 — email —> 7
```
Example (2)

person → name email address
person → name address

name email address
1 → 2 → 3 → 4

address
5

person → name email address
person → name phone address

name email address
1 → 2 → 3 → 4

phone
5

address
6

name email address
1 → 2 → 3 → 45

address

person → name email? address

person → name (email | phone) address

name email address
1 → 2 → 35 → 46

phone

email address
Observation 1: Most of existing approaches infer DTDs or simple XML Schema constructs
- Focus on content models (= regular expressions)
- But:
  - The languages are richer
    - XML Schema, Schematron, RELAX NG
  - Can provide more precise information (optimization)

Observation 2: Evaluation of quality of a schema is not natural

Idea: Exploitation of user interaction
- More natural result
- Exploitation of advanced constructs

Our Approach = SchemaBuilder
User Interaction

- **Problems:**
  - User is an expert in the problem domain
  - Not in the algorithm
  - User is lazy
  - User makes mistakes

- **Two extreme cases:**
  - User provides a setting (weights, amounts, sizes, …)
  - Too general
  - Cannot cover all the cases
    - Context, semantics
  - User participates in every step of the algorithm and chooses from the possibilities
  - Too much work, too many choices

```xml
<bicycles>
  <bicycle>
    <wheel/>
  </bicycle>
  <bicycle>
    <wheel/>
  </bicycle>
</bicycles>
```
STEP 1. Clustering of Elements

- Options:
  - According to element names
  - According to context
  - Path to the root element
  - According to element subtrees

```
<author>
  <name>
    <first>Arthur</first>
    <middle>Conan</middle>
    <last>Doyle</last>
  </name>
</author>
```
```
<book>
  <name>Sherlock Holmes</name>
</book>
```
Our Approach

- **Observation**: XML Schema
  1. Elements in the same context can have different schema
  2. Elements in different contexts can have the same schema

- **Main features**: 
  - We cluster elements according to context (1)
    - Not just element names
  - We cluster according to tree edit distance (TED) between element trees (2)
    - Regardless the context
  - User interaction
  - Type inference
User Interaction

- Two extreme cases:
  - Threshold(s) for similarity of element names, contexts, TED, ...
  - User decides on every initial cluster

- SchemaBuilder:
  - User decisions, but only on subset of clusters
  - Two thresholds: automatic decision $\leq T_1 < \text{user decision} \leq T_2 < \text{do not cluster}$
  - Exploitation of data semantics
  - Reduction of user decisions
Type Inference

- **XML Schema:**
  - Content of element $E$ can be *derived* from content of element $F$
  - Extension – extending the content model
    - Optional/compulsory items
  - Restriction – reducing the amount of instances

- **SchemaBuilder:**
  - Cannot be done automatically in general
  - **Observation:** Inheritance is used, but not often/complex
  - → **User** marks elements with mutually derived types
  - **Main task:** Checking *correctness* of such marking
STEP 2: Schema Generalization

- **Input**: Set of clusters $C_1, C_2, \ldots, C_k$ + marked inheritance

  - PTA + merging state algorithm

- **Output**: A set of schemes $S_1, S_2, \ldots, S_l$

- **SchemaBuilder**: PTA +
  - Inheritance state – preserves the inheritance
  - Permutation state – represents unordered sequence of elements (XML Schema)
  - Group state – globally referenced particle
Merging State Algorithm

- Combinatorial optimization problem (COP)
  - A search space $\Sigma_i$ of solutions (feasible region)
  - Generalizations of $S^{\text{init}}_i$ (PTA)
  - A set $\Omega_i$ of constraints over $\Sigma$
  - Features of XML Schema language
  - Evaluation function $f : \Sigma_i \rightarrow \mathbb{R}^+$ (objective function)
  - MDL (Minimum Description Length) principle
- $\Sigma_i$ is theoretically infinite $\rightarrow$ heuristics $\rightarrow$ suboptimal solution
- ACO (Ant Colony Optimization)
Ant Colony Optimization (ACO)

- Idea: Artificial ants iteratively search space $\Sigma_i$ and improve $S_{init_i}$

- Ant
  - Searches a subspace of $\Sigma_i$ until it “dies”
    - After performing $N_{ant}$ steps
  - Spreads “pheromone”
    - Positive feedback = how good solution it has found so far
    - Negative feedback = how good solution it has found in this iteration
  - Exploits spread pheromone of other ants to select next step
    - Step = a possible way of schema generalization
    - Selected randomly, probability is given by $f$
SchemaBuilder Steps of Ant

- Merging of states
  - sk-strings, kh-kontext, ...

- SchemaBuilder:
  - Permutation substitution
    - Identification of a candidate (sub-automaton) for unordered sequence + checking
  - Group substitution
    - Identification of a candidate for code reuse + checking
  - Checking validity of inheritance

STEP 3: Schema Output

- Converting of the automaton to XML Schema language
- Rules:
  - Classical states $\rightarrow$ regular expression
  - Permutation state $\rightarrow$ unordered sequence
    - all construct of XML Schema
  - Inheritance state $\rightarrow$ derivation of types
    - extension or restriction construct of XML Schema
  - Group state $\rightarrow$ reuse contracts
    - element, attribute, group or attributeGroup constructs of XML Schema
Screenshots
Conclusion

- Advantages of the algorithm:
  - User interaction
    - Simple but powerful
  - Advanced XML Schema constructs
    - Cannot be inferred without UI
- Current work: implementation of a general framework (Inferrer)
- Future work:
  - Transformation of automata to regular expressions
  - Exploitation of negative examples
  - Inference of integrity constraints
  - Inference of grammars


Thank you