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

    $\Rightarrow$ Optimization
    - Storing, querying, updating, compressing, …

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

  $\Rightarrow$ 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

XML Schemas and Grammars

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 α is a regular expression over alphabet $N \cup T$.

Given the alphabet Σ, a regular expression (RE) over Σ is inductively defined as follows:

- $\emptyset$ (empty set) and ε (empty string) are REs
- $\forall\ a \in \Sigma$: a is a RE
- If r and s are REs over Σ, then (rs) (concatenation), (r|s) (alternation) and (r*) (Kleene closure) are REs

- DTD adds: (s|ε) = (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 $\Rightarrow$ 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 $E_1, E_2, \ldots, E_n$ we create production $E \rightarrow E_1\ E_2 \ldots E_n$
2. Clustering of rules of IG
   - 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)

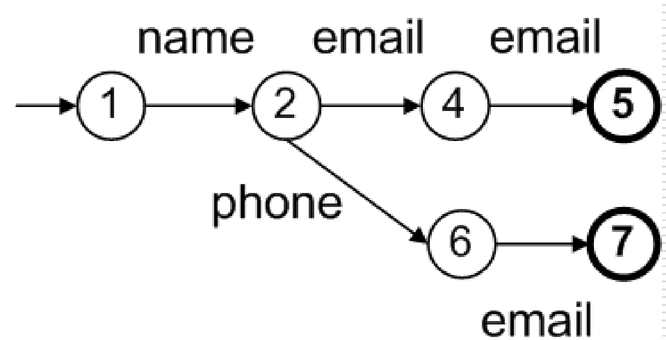
```
...
<person id="123">
  <name>
    <first>Irena</first>
    <surname>Mlynkova</surname>
  </name>
  <email>irena.mlynkova@gmail.com</email>
  <email>irena.mlynkova@mff.cuni.cz</email>
</person>
<person id="456" holiday="yes">
  <name>
    <surname>Necasky</surname>
    <first>Martin</first>
  </name>
  <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

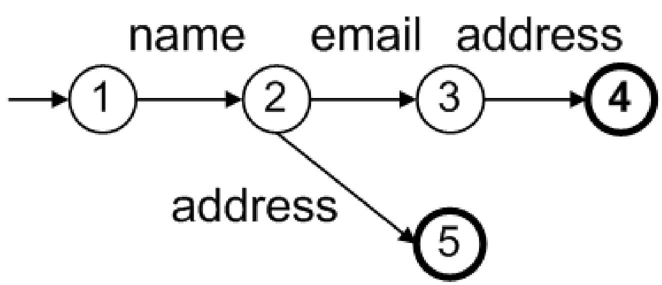
first → PCDATA
surname → PCDATA
email → PCDATA
phone → PCDATA

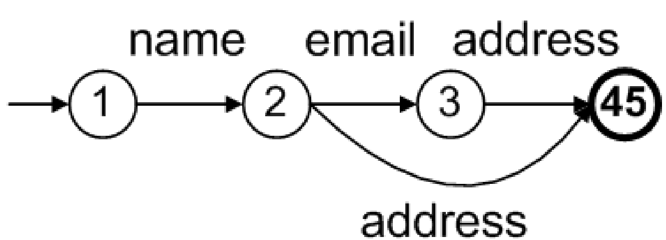
person:



Example (2)

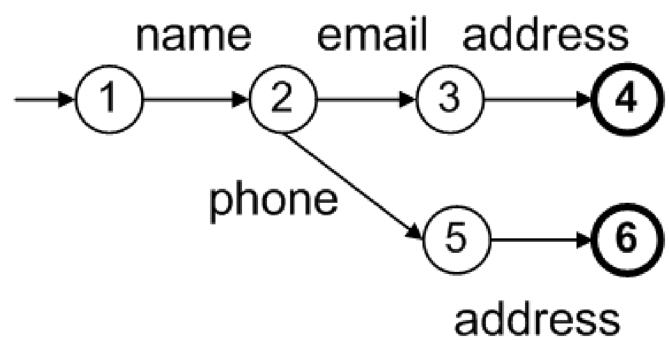
person → name email address
person → name address

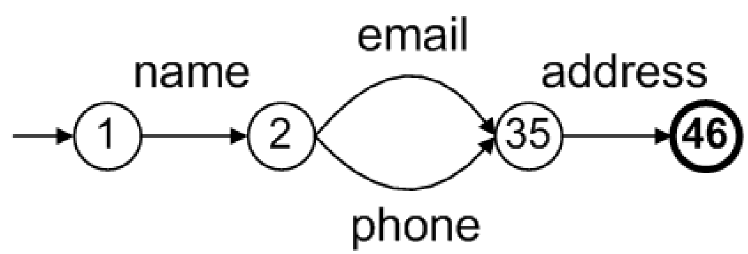




person → name email? address

person → name email address
person → name phone 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_{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

E → A B C C C
E → A B C C

E → A (B | X) C+

E → A B C C C
E → A B C C

E → A B C+ X

- 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 s.t. d is not valid against S_{orig}
  - Aim: to find schema $S_{correct}$ = correction of S_{orig} s.t.
    - For $\forall$ d $\in$ D : d is valid against $S_{correct}$ and
    - $dist(S_{orig}, S_{correct}) \leq dist(S_{orig}, S)$ for $\forall$ S $\in$ $\Sigma_{correct}$; where $\Sigma_{correct}$ is the set of all possible corrections of S_{orig}

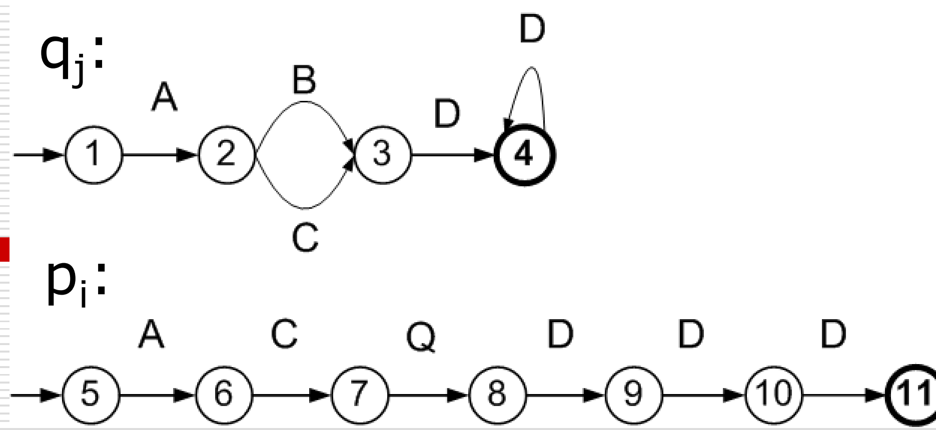
  $\Rightarrow$ Output: $S_{correct}$
- Step 2: Specialization of the input schema
  - Assumption: $\forall$ d $\in$ D : d is valid against $S_{correct}$
  - Aim: to specialize REs in $S_{correct}$ with regard to D

  $\Rightarrow$ Output: $S'_{correct}$

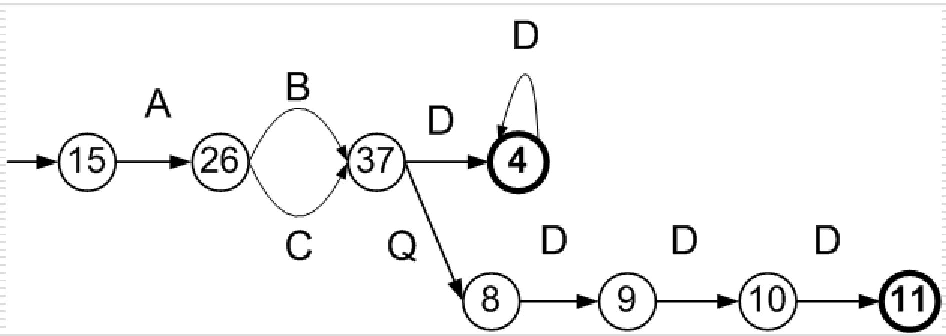
Step 1. Schema Correction

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

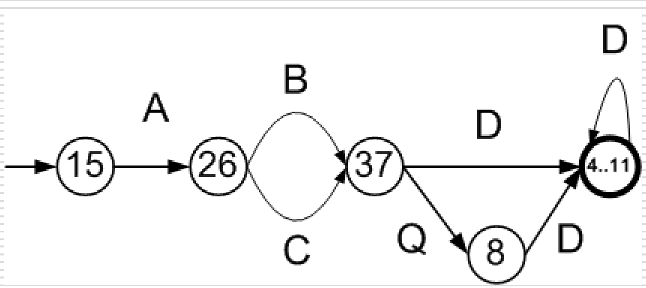
Example



PTA:



Merged:



Merging State Strategy

- Combinatorial optimization problem (COP)
  - A search space Σ of solutions (feasible region)
  - A set Ω of constraints over Σ
  - Evaluation function $f : \Sigma \rightarrow R_0^+$ (objective function)
- Our case:
  - Σ = a set of possible generalizations of input schema S_{input}
  - Ω is given by the features of XML schema language
  - f = 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: Σ 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 Σ and improve S_{input}
- Ant
  - Searches a subspace of Σ until it "dies"
    - After performing N_{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 is valid against $S_{correct}$
- Aim: to specialize REs involved in $S_{correct}$ with regard to D $\Rightarrow$ Output: $S'_{correct}$
- Idea: Parsing of documents in D and checking preciseness of $S'_{correct}$
- 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: $\forall\, e \in S_{correct}$ we set usage flag $\varphi_{used}(e)$

$$\begin{array}{lllllll} E & \rightarrow & A & C & & & \\ E & \rightarrow & A & B & B & B & C \\ E & \rightarrow & A & B & C & & \\ E & \rightarrow & A & B & B & B & B \end{array} \qquad E \rightarrow A\ B{\star}\ C?\ Q?$$

$$\begin{array}{cccccc} E & \rightarrow & A & B{\star} & C? & Q? \\ & & T & F & T & F \end{array}$$

$$\begin{array}{cccccc} E & \rightarrow & A & B{\star} & C? & Q? \\ & & T & T & T & F \end{array}$$

- Note: Elimination of unused schema fragments preserves correctness of content models

$$E \rightarrow A\ B{\star}\ C?$$

2. Correction of Lower and Upper Bounds

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

$$\begin{array}{lll} E & \rightarrow & A \\ E & \rightarrow & B \\ E & \rightarrow & A\ A \end{array}$$

$$E \rightarrow A+ \ | \ B \ | \ (C\ D)$$

	$E \rightarrow$	$A+$	B	$(C\ D)$
Start:	φ_{min}	∞	∞	∞
	φ_{max}	0	0	0
$E \rightarrow A$	φ_{min}	1	0	0
	φ_{max}	1	0	0
$E \rightarrow B$	φ_{min}	0	0	0
	φ_{max}	1	1	0
$E \rightarrow A\ A$	φ_{min}	0	0	0
	φ_{max}	2	1	0

- Note: φ_{min}(e) and φ_{max}(e) cover φ_{used}(e)

3. Correction of Operators
4. Factorization

$a?, b? \rightarrow (a, b)?$
$a?, b^* \rightarrow (a, b^+)?$
$a?, b? \rightarrow a \mid b$
$a?, b^* \rightarrow a \mid b^*$

$a?? \rightarrow a?$
$a^{++} \rightarrow a^+$
$a^{**} \rightarrow a^*$
$a^*? \rightarrow a^*$
$a?^* \rightarrow a^*$
$a^{+*} \rightarrow a^*$
$a^{*+} \rightarrow a^*$
$a?^+ \rightarrow a^*$
$a^+? \rightarrow a^*$
$aa^* \rightarrow a^+$
$a^+a^* \rightarrow a^+$
$a?a^+ \rightarrow a^*$
$(ab) \mid (ac) \rightarrow a(b \mid c)$

- Restriction of content models
  - Can be applied only if validity is not violated
- Improving readability, simplification of structure
  - Classical step

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 $S_{correct}$
- In the worst case: S_{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 $\Rightarrow$ mutual comparison of impact
  - Exploitation
    - Storage strategies of XML data
  - Further improvements
    - User interaction, inference of integrity constraints, other schema languages (RELAX NG, Schematron)…

Thank you