Even an Ant Can Create an XSD

Ondrej Vosta,

Irena Mlynkova, Jaroslav Pokorny

ondra.vosta@centrum.cz,
irena.mlynkova@mff.cuni.cz, jaroslav.pokorny@mff.cuni.cz

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

Overview

1. Introduction
2. Existing approaches
3. Proposed approach
4. Conclusion

Introduction (1)

- 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, …

Introduction (2)

- 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 schemes at all
  - Schema = a kind of documentation
    - Documents are not valid, schemes are not correct
  - XML Schema language is not used
    - Too complex, too difficult

Mlynkova, Toman, Pokorny: Statistical Analysis of Real XML Data Collections. COMAD 2006, New Delhi, India. Tata McGraw-Hill Publishing Co. Ltd., 2006. ISBN 0-07-063374-6.

Introduction (3)

- **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 schemes are too general (*, +, recursion, …) $\Rightarrow$ S_D can be more precise**

Overview

1. Introduction
2. Existing approaches
3. Proposed approach
4. Conclusion

Existing Approaches

- **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 a set of regular expressions**
  - **Gold's theorem: Regular languages are not identifiable only from positive examples (XML documents)**

    $\Rightarrow$ other information, heuristics, subclass of languages
- **Problem:**
  - **Most of existing approaches infer DTDs**
  - **Single exception: Bex et al. – VLDB'07**
    - **Focus on context of elements and constructs used in real-world data**

Our Approach

- Exploitation and combination of the best of existing approaches
  - Verified techniques can be re-used
- Focus on purely XML Schema constructs
  - New functionality is added
  - E.g. unordered sequences, elements with same name, but different structure
- General and extensible algorithm
  - New functionality can be added in future

⇒ Aim:
  - More realistic XML schemes
  - Increase of popularity and exploitation of XML Schema

Overview

1. Introduction
2. Existing approaches
3. Proposed approach
4. Conclusion

Motivating Example (1)

- DTD: All elements at the same level
- XSD: Globally vs. locally defined elements

⇒ Elements with same name but different structure

```
<author>
  <name>
    <first>Arthur</first>
    <middle>Conan</middle>
    <last>Doyle</last>
  </name>
</author>
```

```
<book>
  <name>Sherlock Holmes</name>
</book>
```

Existing approaches would infer a common schema

Motivating Example (2)

- **Ordered sequence:**
  - **DTD: `(a,b,c)`**
  - **XSD: element sequence**
    - **Ordered sequence of subelements**
- **Unordered sequence:**
  - **DTD: `((a,b,c)|(a,c,b)|(b,a,c)|(b,c,a)|`**
    `(c,a,b)|(c,b,a))`
    - **For *n* elements *n*! possible sequences**

  - **XSD: element `all`**
    - **Unordered sequence of subelements**
- **No increase in expressive power**
  - **Syntactic sugar, but important!**

Overview of the Proposed Algorithm

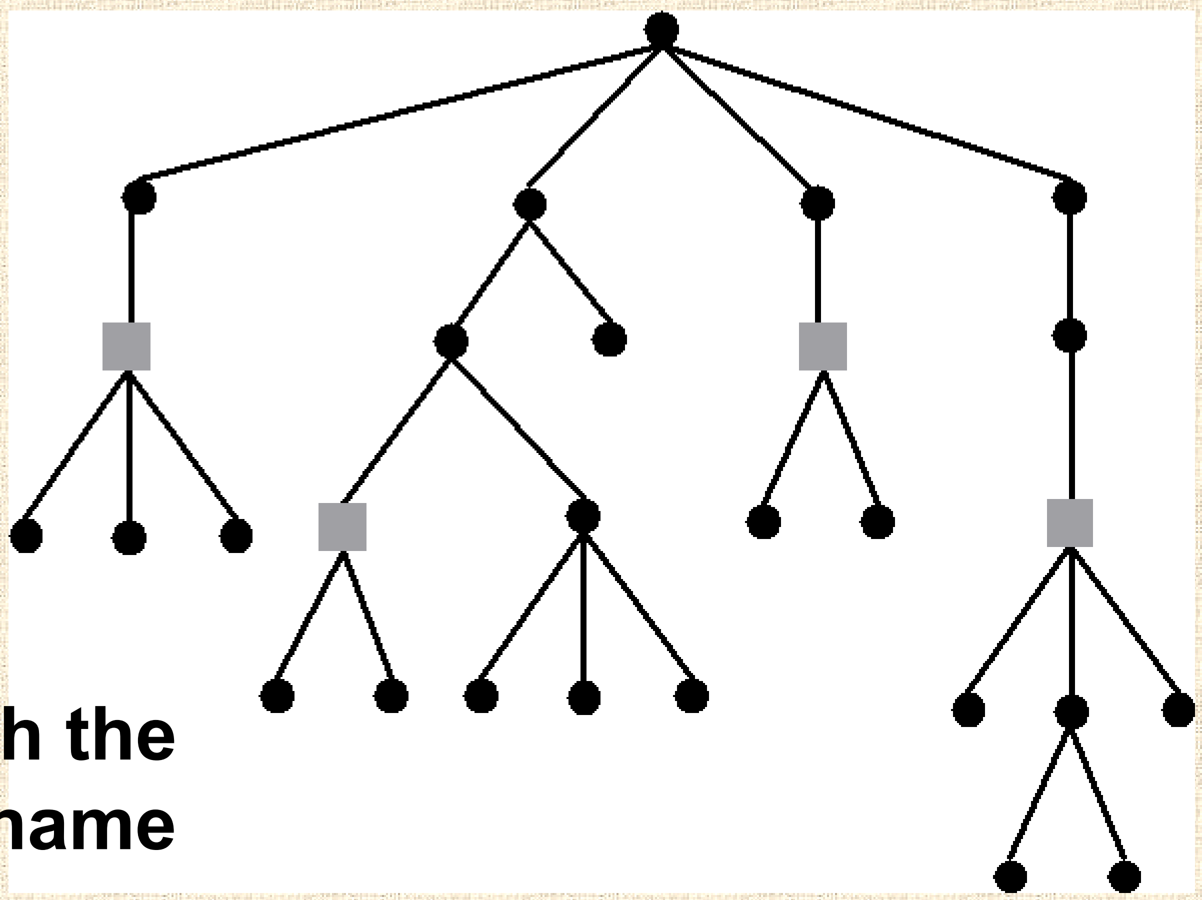
- A heuristic approach
- Steps:
  1. Clustering of elements with common schema
     - We improve the clustering to be XSD-aware
  2. Schema generalization within the clusters
     - We combine existing approaches + add inference of XML Schema constructs
  3. Rewriting of generalized schema into XSD syntax
     - We output XSDs with true XML Schema constructs
       - We are able to find them in 1. and 2.

Clustering of Elements

- **Aim: Support of elements with the same name but different structure**
  - **Existing works: Clustering of all elements with the same name**
- **Idea: Clustering of elements with the same context and similar structure**

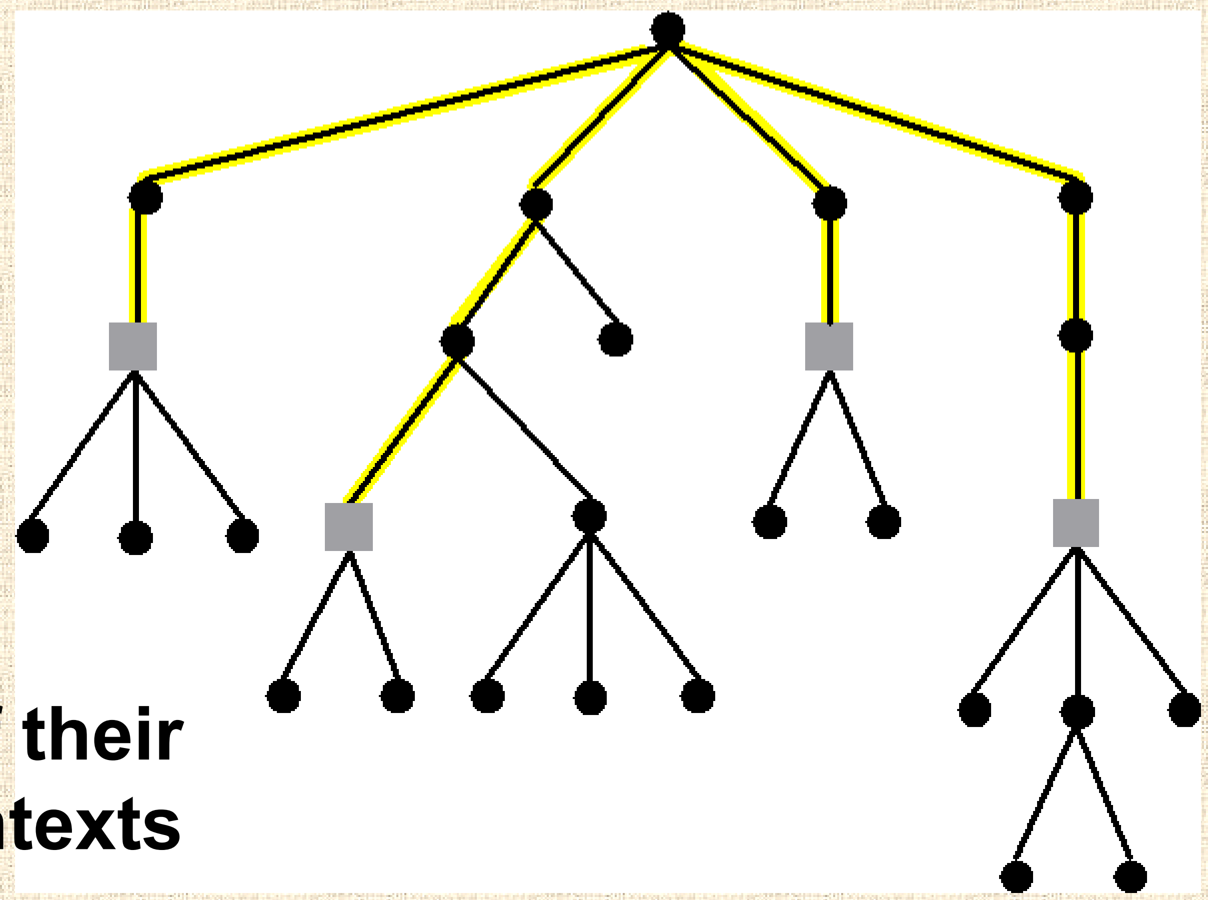
1. **XML documents D_1 and $D_2 \Rightarrow$ trees T_1 and T_2**
2. **Clustering of elements on the basis of context**
   - **Path from root node**
3. **Clustering of elements on the basis of similarity of element trees**
   - **XML-aware tree edit distance**

Clustering of Elements: Step 1



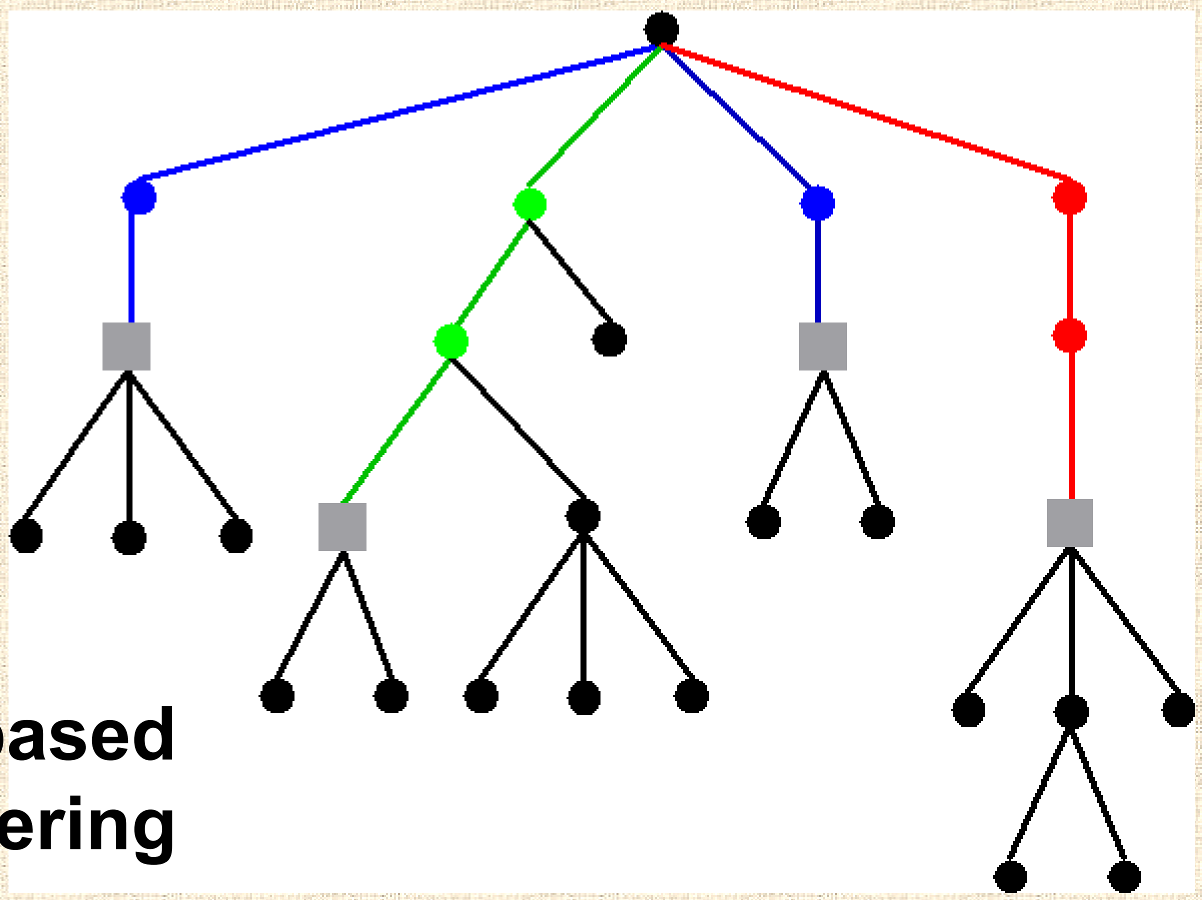
Elements with the same name

Clustering of Elements: Step 2



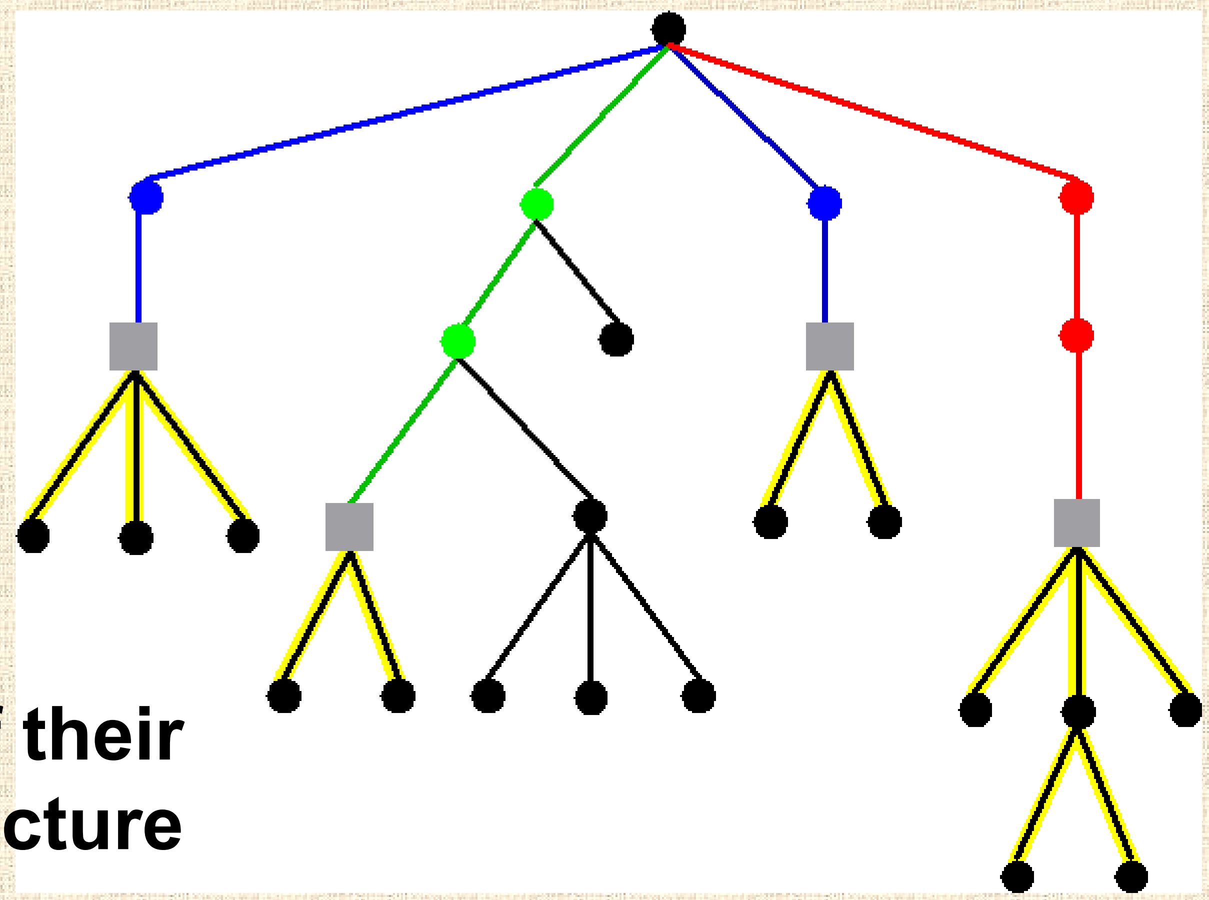
Analysis of their contexts

Clustering of Elements: Step 3



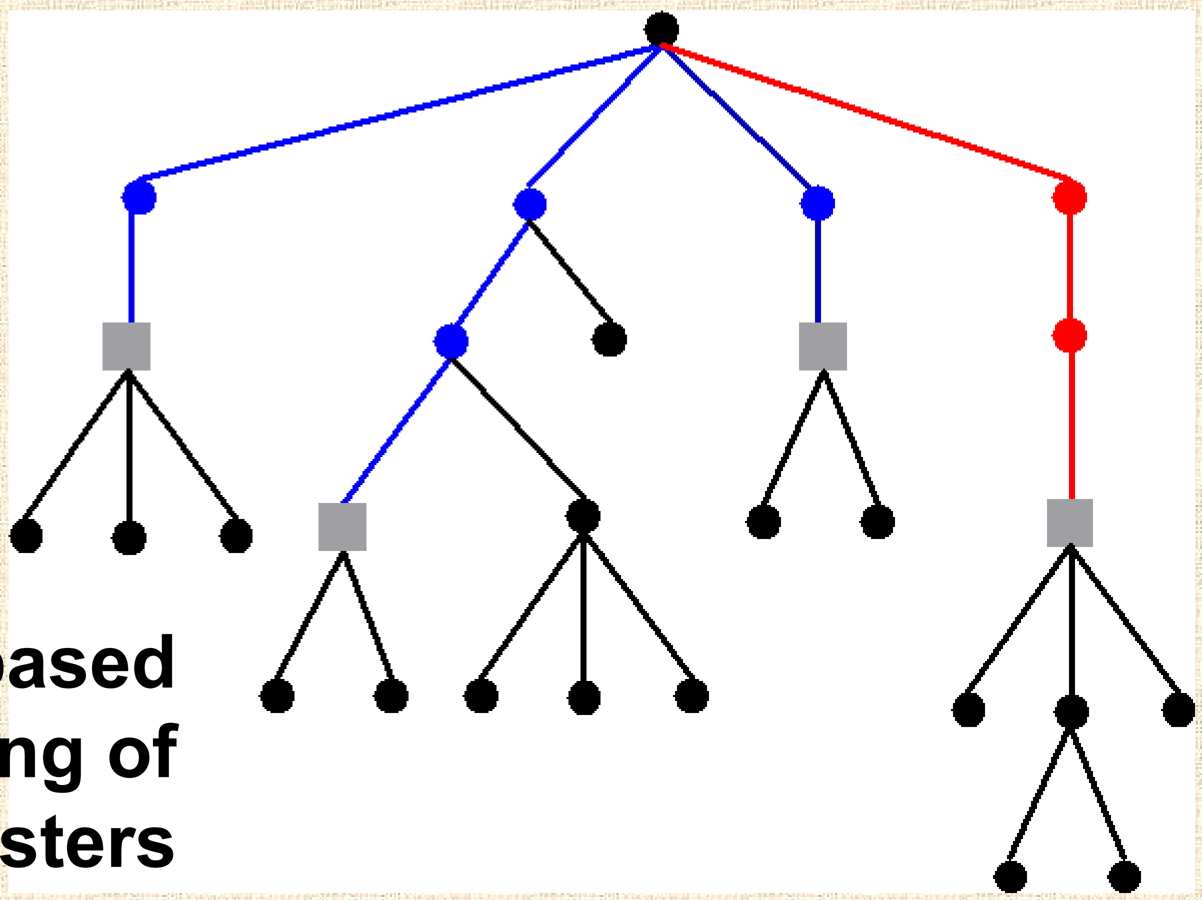
Context-based clustering

Clustering of Elements: Step 4



Analysis of their structure

Clustering of Elements: Step 5



Structure-based merging of clusters

Schema Generalization (1)

- **Input: A set of clusters $C_1, C_2, \ldots, C_k$**
  - **Cluster = a set of elements $C_i = \{e_1, e_2, \ldots, e_n\}$ with common schema to be searched**
- **Output: A set of schemes $S_1, S_2, \ldots, S_k$**
- ***Idea:***
  - **S_i^{init} = s simple schema accepting only elements from C_i**
  - **S_i^{init} is generalized until a "reasonable" schema is found**
    - **Theoretically infinite number of possibilities**

$\Rightarrow$ Combinatorial optimization problem (COP)

- **A search space Σ_i of solutions (feasible region)**
- **A set Ω of constraints over Σ_i**
- **Evaluation function $f : \Sigma_i \rightarrow R_0^+$ (objective function)**

Schema Generalization (2)

- Input: cluster C_i
- Σ_i = a set of possible generalizations of S_i^{init}
- Ω is given by the features of XML Schema language
  - We omit attributes (for simplicity)
  - DTD operators: , | + * ?
  - XSD operator: & (= element `all`)
- f = evaluates the quality of given $S \in \Sigma_i$
  - MDL (Minimum Description Length) principle
- Search algorithm: ACO (Ant Colony Optimization)
  - Σ_i is theoretically infinite $\Rightarrow$ heuristics $\Rightarrow$ suboptimal solution

Ant Colony Optimization (ACO)

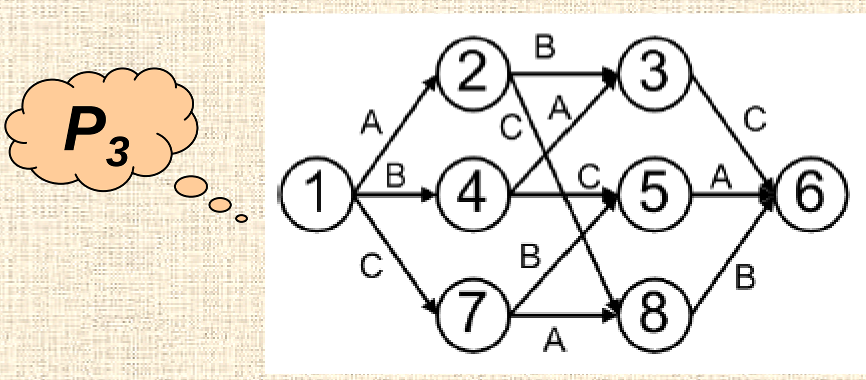
- Meta-heuristics for solving *COPs*
- Idea: Artificial ants iteratively search space Σ_i and improve S_i^{init}
- Ant
  - Searches a subspace of Σ_I until it “dies”
    - After performing N_{ant} steps
  - Spreads “pheromone”
    - Positive feedback = how good solution it has found so far
    - Negative 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*

Possible Steps

```
name → #PCDATA
name → first middle last
```

- Each element in C_i = simple production rule of a grammar ⇒ prefix-tree automaton
- Idea: Generalization = merging states of automaton
- 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."
- Our improvement: Inferring of & operators
  - Unordered sequences

Inferring of & Operator



- **Observation: Complexity of unordered sequences is limited**
  - **XML Schema 1.0: Unordered sequence and its elements have occurrence (0,1)**
  - **XML Schema 1.1: Unordered sequence has occurrence (0,1)**
- **Idea:**

1. **First level candidates = subgraphs having one input node n_{in} and one output node n_{out} and out-degree(n_{in}) > 1**
2. **Second level candidates = enough similar to P_n**
   - **P_n = automaton that accepts permutation of *n* elements**
   - **Simplified tree-edit distance – P_n is highly restrictive**
3. **Observation: Permutation of *n* items contains permutations of *n* – 1 items $\Rightarrow$ candidates are sorted and extended**

Evaluation of Steps – MDL Principle

- Input: Current schema S_i^x and its generalization S_i^y
- Output: step(S_i^x, S_i^y) = evaluation of this step

$step(S_i^x, S_i^y) = f(S_i^x) - f(S_i^y) + pos(S_i^x, S_i^y) + neg(S_i^x, S_i^y)$

  - $pos(S_i^x, S_i^y) \geq 0$ = positive feedback
  - $neg(S_i^x, S_i^y) \leq 0$ = negative feedback
  - f = objective function
- MDL principle:
  - Good schema is enough general ⇒ low number of states of automaton
  - Good schema preserves details ⇒ express instances using short codes
    - Most of the information is carried by the schema

⇒ *f* evaluates the size of schema and size of production rules to derive the instances

Advantages and Results

Advantages:

- ACO ⇒ any rules to produce steps of an ant
- MDL ⇒ evaluates any schema no matter how inferred

⇒ The approach can be easily extended

Results:

- **Permutations**

Size of the set	Percentage of permutations 10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
3	no	partly	no	partly	no	no	partly	yes	yes	yes
4	no	yes	partly	yes	partly	partly	partly	yes	yes	yes
5	no	no	partly	partly	partly	partly	partly	yes	yes	yes

- **Elements with different content**
  - **The clustering depends on the threshold of similarity metric**
  - **50% of similarity seems to be reasonable for real-world data**

Overview

1. Introduction
2. Existing approaches
3. Proposed approach
4. Conclusion

Conclusion

- Advantages of algorithm:
  - Inspiration in verified approaches
  - Support of new XML Schema constructs
  - Extensibility
- Future work
  - Focus on more XML Schema constructs
    - Element groups, attribute groups, inheritance, …
    - More realistic result vs. syntactic sugar
  - User interaction
    - User-specified clusters, negative examples, influence on steps of ants, selection of required constructs, …

Thank you