Using AutoMed for XML Data Transformation & Integration

L. Zamboulis - A. Poulovassilis

{lucas,ap}@dcs.bbk.ac.uk

Overview

- Objective: transformation & integration of XML files
  - Schema matching process assumed
  - Framework built within the AutoMed system

Motivation

- Interoperability
- Related work on relational databases
- Need for XML-specific solutions

Example Applications

- ❑ XML-enabled web services & applications
- ❑ P2P applications
- ❑ XML Messaging

Aims

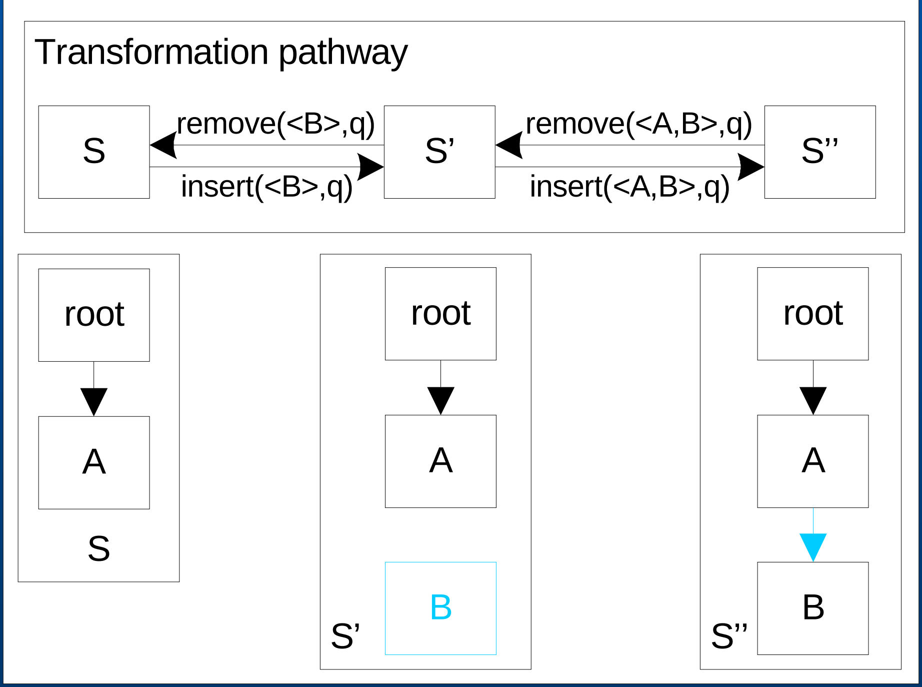
- XML-specific solution:
  - Insert-remove-rename operations on elements, attributes, edges
  - Efficient ‘move’ operation
  - Element-to-attribute, attribute-to-element transformations
- Ability to create synthetic structure, to avoid loss of data due to structural incompatibilities
- Automation

Problems

- Schema matching process is semi-automatic
- Due to the nature of XML:
  - Ordering policy
  - Due to mixed elements, process might be semi-automatic

AutoMed System

- Graph environment (HDM)
- Schema-based transformation approach
- Automatically derivable reversible transformations



XML DataSource Schema

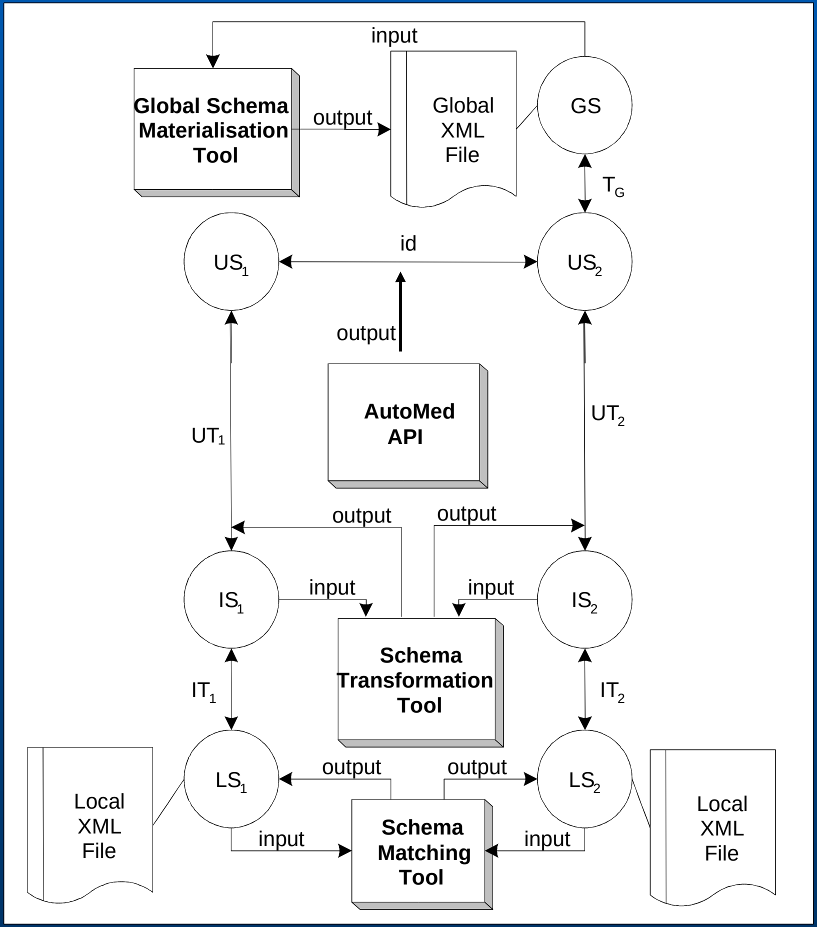
- Basic characteristics
  - Automatically derived from an XML file
  - Structure-only representation
  - XML format: ease of traversal & manipulation
- Comparison to
  - DTD
  - XML Schema
  - DataGuides

General Integration Scenario

- Transform LS_i to US_i, using automatically reversible transformations
- Id transformations
- Create GS from arbitrary US_i

XML Integration Scenario

- Schema matching phase
- Schema transformation phase
- id phase
- Global schema materialisation



Schema Matching

- Types: 1-1, 1-n, n-1, n-m
- Example: 1-n match

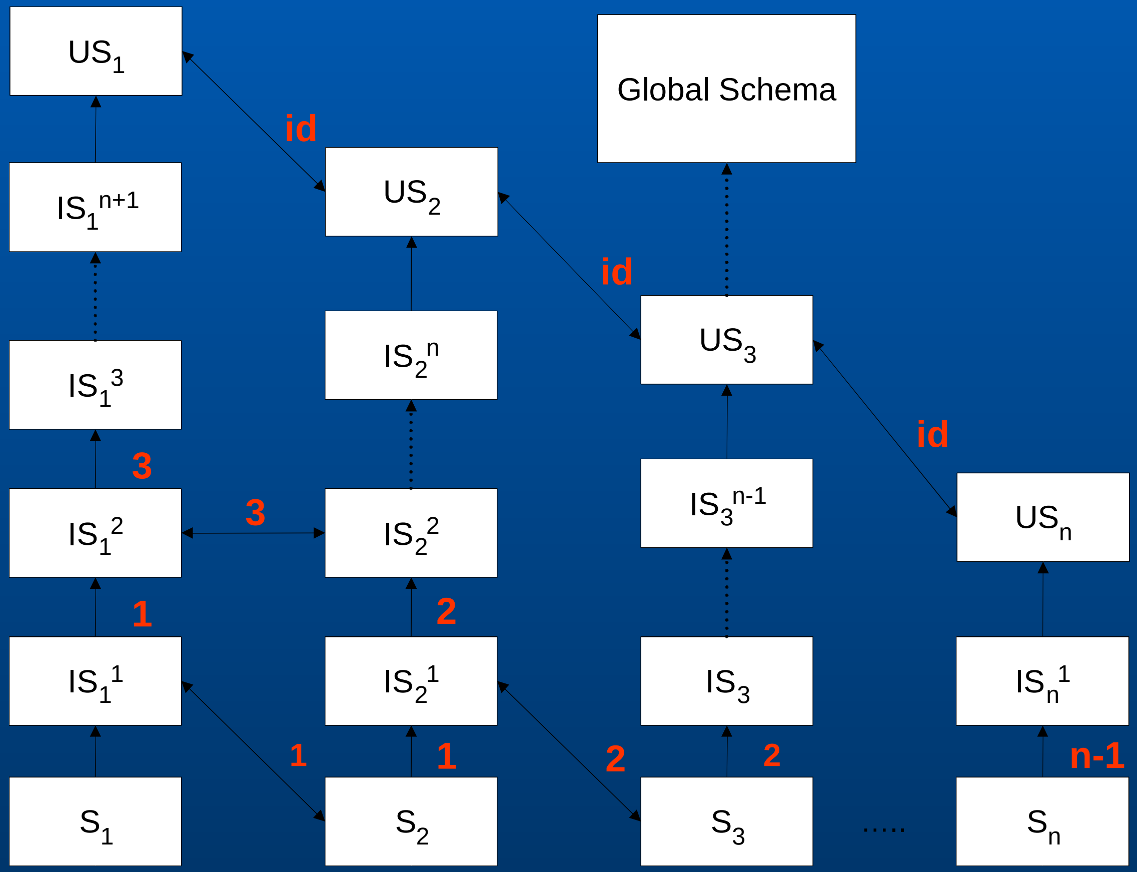
  S_1: <author dob="1965-07-15"/>

  S_2: <author day="15" month="07" year="1965"/>
- Necessary transformations:
  - add attributes day, month, year in S_1
  - delete attribute dob from S_1
- The reverse transformation pathway describes a n-1 match.

Schema Transformation (1/2)

- Global schema GS is not given
- Local schemas LS_2 – LS_n are transformed to match the structure of LS_1
- All local schemas are extended to contain the elements, attributes and text nodes of the other schemas

Schema Transformation (2/2)



Transformation Algorithm (1/2)

- Growing phase: traverse the target schema and issue an add/extend transformation for every construct that does not exist in the source schema.
- Shrinking phase: traverse the source schema and issue an delete/contract transformation for every construct that does not exist in the target schema.

Transformation Algorithm (2/2)

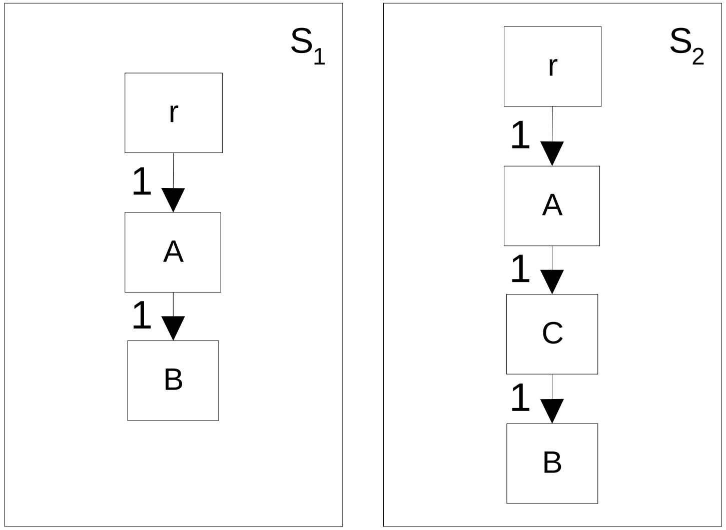
- Insert/remove text nodes:
  - If a source element E_S has n text nodes and its corresponding element E_T has m text nodes, with $n,m>0$ and $n \neq m$, process is semi-automatic

Transformation Types

- HDM (graph) level:
  - Insert: add or extend
  - Remove: delete or contract
  - rename
- Schema level:
  - Insert, remove or rename schema constructs
  - Move element/subtree
  - Element ↙ attribute
  - Attribute ↙ element

Transformation Example (1/4)

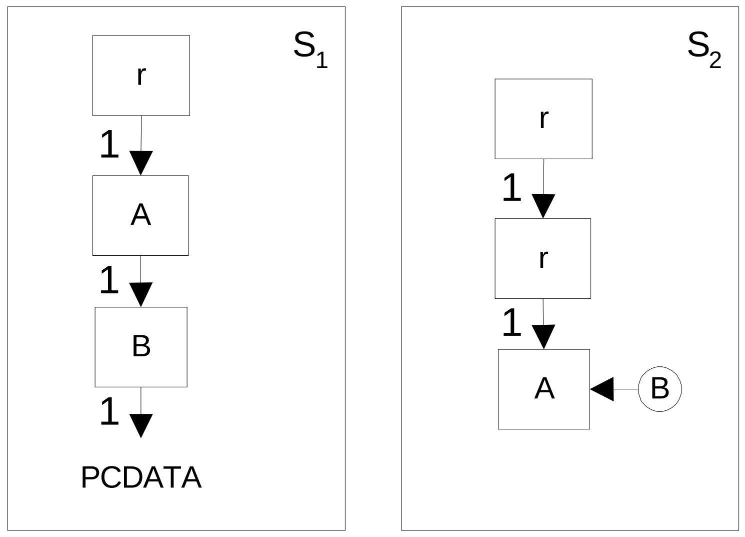
- Insert element C
  - extend(<C>,null)
  - extend(<A,C>, null)
  - extend(<C,B>, null)
- Remove element C
  - contract(<A,C>, null)
  - contract(<C,B>, null)
  - contract(<C>,null)



Insert/remove element

Transformation Example (2/4)

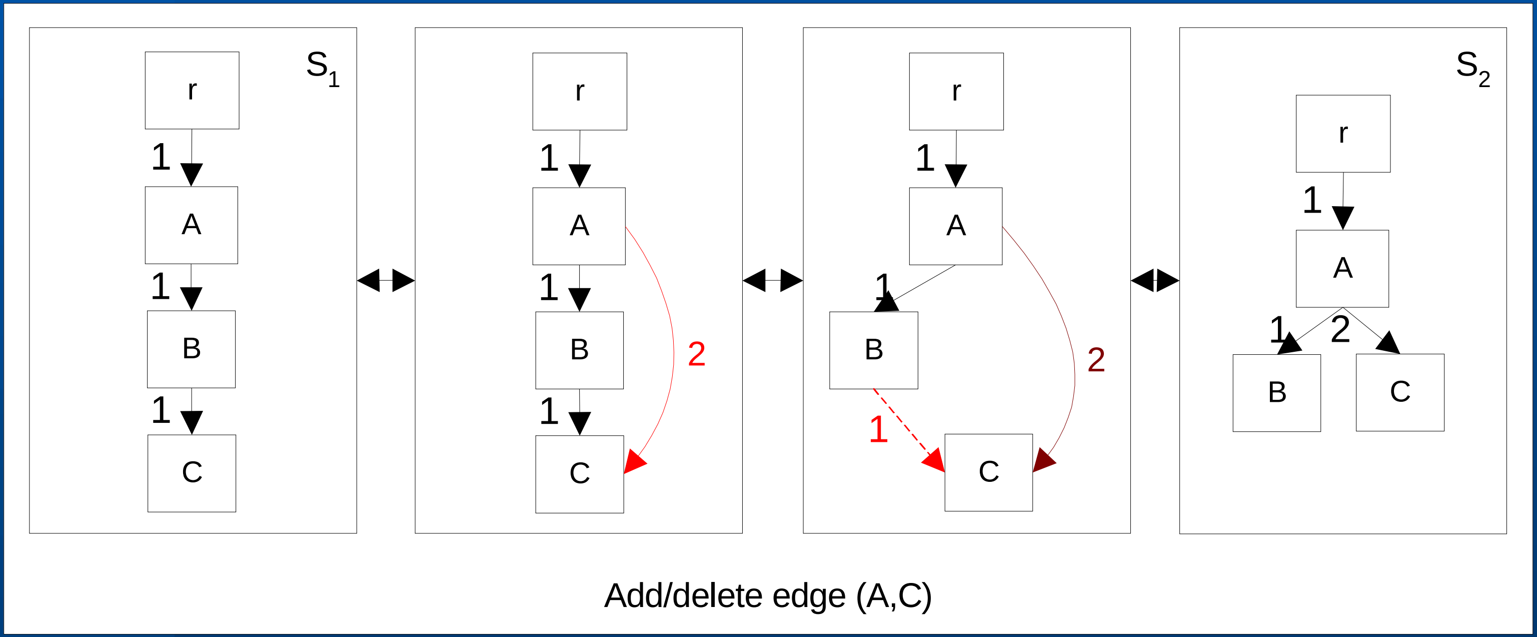
- Element-to-attribute transformation
  - insert(<A:B>,q)
  - remove(<A,B>,q)
  - remove(<B,PCDATA>,q)
  - remove(<B>,q)
- Attribute-to-element
- transformation
  - insert(<B>,q)
  - insert(<A,B>,q)
  - insert(<B,PCDATA>,q)
  - remove(<A:B>,q)



Element↗↙Attribute

Transformation Example (3/4)

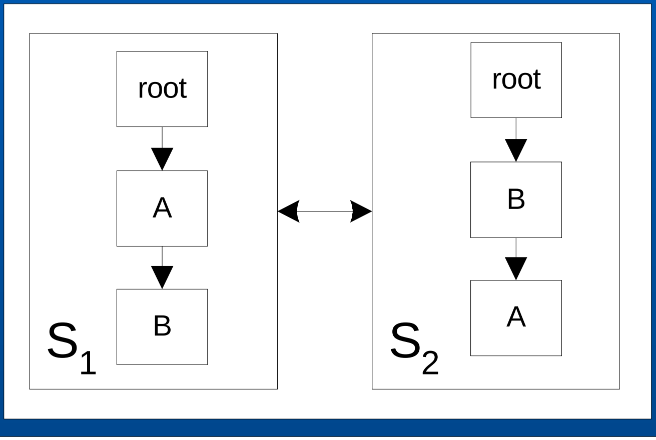
- **Insert/remove edge: move operation**

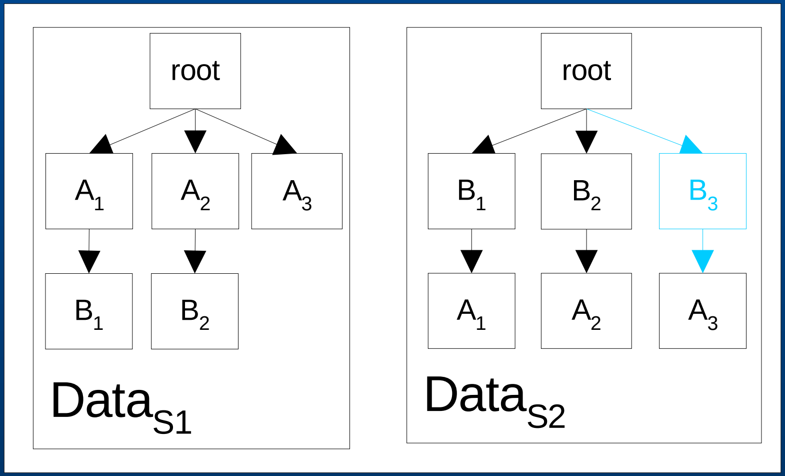


Add/delete edge (A,C)

Transformation Example (4/4)

- Insert/remove edge: move operation
- Pathway:
  - extend(<B>,q)
  - extend(<root,B>,q)
  - extend(<B,A>,q)
- Extend because we create synthetic data





Global Schema Materialisation

- **Strategy:**
  - **Materialise root and its attributes**
  - **Consider all edges (e_p,e_c) in a depth-first way**
  - **Materialise e_c and its attributes**

Conclusions

- XML specific solution:
  - element↗↙attribute transformations
  - move operation
- No loss of data by synthetically creating missing structure.

Future Work

- Include more types of XML data sources, e.g. XML Native DBs
- Streaming integration and materialisation
- Targeted schema evolution
- Targeted rematerialisation of GS

Resources

- http://www.doc.ic.ac.uk/automed