University of Washington
Department of Computer Science & Engineering

# Managing XML and Semistructured Data

Lecture 15: Query Analysis

Prof. Dan Suciu

Spring 2001

## In this lecture

- Query rewriting
  - examples
- Query rewriting with schema

Resources

- *Optimizing Regular Path Expressions Using Graph Schemas*, M.Fernandez and D.Suciu, Data Engineering, 98
- *Query Optimization for Structured Documents Based on Knowledge on the Document Type Definition,* K. Bohm, K. Gayer, K. Aberer, T. Özsu

## Query Analysis

Generic term to describe:

- Query rewriting based on schema information
- Query containment and minimization

## Query Rewriting

Problem:

- Given a query Q
  - Regular path expression
  - Or more complex Xquery expression
- Given a schema S
  - graph schema
  - DTD
  - XML-Schema
- Rewrite Q to some $Q^S$ s.t.
  - Q is equivalent to $Q^S$ over databases conforming to S
  - $Q^S$ is more efficient than Q

## Query Rewriting

*Optimizing Regular Path Expressions Using Graph Schemas*, M.Fernandez and D.Suciu, Data Engineering, 98

Simplest setting:

- Regular path expression
- Graph schemas

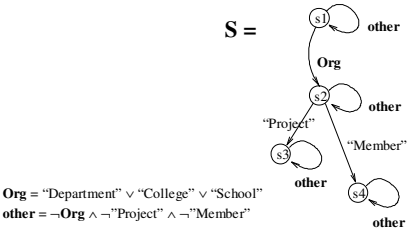
## Example of Query Rewriting

Q = //Department//Project

- Naive evaluation: need to traverse entire graph (or tree)

## Example of Query Rewriting

Graph Schema:




## Example of Query Rewriting

- Schema says: "there can be at most one Department edge; below, there can be at most one Project edge"

Q = //Department//Project

$Q^S$ = (**other**)*/Department/(**other**)*/Project

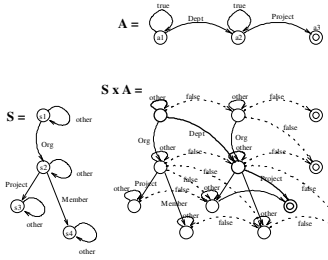
**other** = ¬ "Department" ∧ "College" ∧ "School" ∧ ¬"Project" ∧ ¬"Member"

- $Q^S$ can be evaluated more efficiently than Q
  - Why ?

## Example of Query Rewriting

- How to construct $Q^S$ systematically from Q and S ?
- Step 1 build the automaton A for Q
- Step 2 build the product automaton S x A
- Step 3 $Q^S$ = expression of S x A

## Example of Query Rewriting




$Q^S$ = (**other**)*/Department/(**other**)*/Project

## Query Rewriting

Correctness:

**Proposition** If the instance I conforms to S, then Q(I) = $Q^S$(I)

That is, Q and $Q^S$ are equivalent over databases conforming to S

## Query Rewriting

Efficiency

- Given query Q, instance I, define:
  cost(Q,I) = | {w(I) | w∈ prefix(Lang(Q))} |

**Proposition** If Q and Q' are equivalent over all databases conforming to S, and if I conforms to S, then cost($Q^S$,I) ≤ cost(Q',I)

Hence, $Q^S$ is optimal (in a certain sense)

## Query Rewriting

*Query Optimization for Structured Documents Based on Knowledge on the Document Type Definition,* K. Bohm, K. Gayer, K. Aberer, T. Özsu

More complex settings:

- Schema = DTD
- Query = region algebrar (think: Xpath)

Problem is more complex; this works proposes some solution

## Query Rewriting

Idea: analyze DTD and extract 3 relations:

**Exclusivity**. Element is E1 exclusively contained in E2 if every path from the root to E1 goes through E2

Xpath simplification:
E1[ancestor-or-self::E2] ➔ E1

## Query Rewriting

**Obligation** E1 obligatorily contains E2 if it has a child of type E2

E1[E2] ➔ E1

## Query Rewriting

**Entrance Location** E is an entrance location for E1, E2 if every path from E1 to E2 goes through some E

E1[ancestor-or-self::E2] ➔
E1[ancestor-or-self::E[ancestor-or-self::E2]]

## Query Rewriting

Add these rules, plus variations, to a rule-based optimizer

- HyperStorM – a Structured Document Database
- On top of VODAK – an oo database system

Open question: does this approach exploit all the information in a DTD/XML-Schema ? How can we exploit what is not used ?