Advanced XML / Data on the Web

Lecture 1

Lars Birkedal [birkedal@it-c.dk]

The IT University of Copenhagen

Welcome to AXML

Teacher: Lars Birkedal

- Cand. Scient. from University of Copenhagen, 1994
- Ph.D. from Carnegie Mellon University, 1999
- Started at ITU March 2000 as Assistant Professor
- Now Associate Professor and Head of Theory Department
- Research Interests: Applications of mathematical logic and category theory to computer science, especially concerning the semantics of programming languages and type theories. I am also interested in the implementation of advanced programming languages and co-developer of The ML Kit and SML-Mix.

Outline of this lecture

- Goals of the course
- Prerequisites
- Resources
  - textbooks
  - research papers
- Course Format
- Overview of the course

Goals of the Course

XML has emerged as the de facto standard for data interchange on the web.
New challenges to programming languages, applications, and database systems
Upon completion of this course you should be able to

- identify and understand some of the major challenges offered by XML as seen from the database and programming language perspectives
- understand some of the solutions suggested by the research community to these challenges

Glimpse at current XML standards and technology.

Prerequisites

- Webprogramming course
- Databases
- Introduction to algorithms
- Some advanced programming language course

Resources

Textbooks

- **Data on the Web: from Relations to Semistructured Data and XML by Abiteboul, Buneman, Suciu**
  - for foundations
- W3C homepage, `www.w3.org`
  - for current standards
- Professional XML Databases by Kevin Williams
  - for current XML technologies

Research Papers

- a selection of about 20 research papers from recent conferences and journals

Course Format

- an advanced course, not offered before, new topic for everyone
- you are the “front-runners”
- lectures Tuesdays 9-12 in room 1.90
- lectures cover essential points, read the rest on your own
- grading: pass / no-pass based on 3-4 mandatory assignments
- assignments can be handed in by groups of 1–4 persons
- course home page: will be up next week
- slides and research papers will be available at course home page

Course Overview

Semistructured Data

- basic model (edge-labelled graphs)
- graph bisimulation
- computing graph bisimulation

Course Overview

XML Data Model, DTDs, and XSLT

- Review of XML data model (node-labelled graphs)
- DTDs
- Review of XSLT and a formal semantics of patterns in XSLT

Course Overview

Programming with XML

- Review of DOM
- SAX

Course Overview

Query Languages

- for semistructured data
  - Path expressions
  - Lorel
  - UnQL
- for XML
  - XML-QL
  - XQuery

Course Overview

Types I: schemas for unordered data

- Subsumption for XML Types
- Extracting Schemas from Semistructured Data

Course Overview

Types II: schemas for XML

- XML Schema
- DSD: A Schema Language for XML

Course Overview

Types III: XDuce and Regular Expression Languages

- XDuce: A Typed XML Processing Language
- Regular Expression Types for XML
- Regular Expression Pattern Matching for XML

Course Overview

- Types IV: Taxonomy of XML Schema Languages
- Constraints and Keys

Course Overview

Automata Theory for XML

- Automata- and Logic-based Pattern Languages for Tree-Structured Data

Course Overview

Semistructured Data and Mobile Computation

- Mobile Ambients
- Ambients and semistructured data

Course Overview

Not covered:

- systems (Ch. 8–11 in course book): read on your own
- query analysis
- indexes
- . . .

Semistructured Data

- basic model (edge-labelled graphs)
- graph bisimulation
- computing graph bisimulation

Resources:

- ABS: Chapter 2 + Section 6.4.3
- Supplementary: Buneman et. al.: Adding Structure to Unstructured Data (will be available from course home page)

Semistructured Data (ssd)

Aka: “schemaless” / “untyped” / “self-describing” data

Syntax for ssd (example):

```
{paper: {author: "Abiteboul",
         author: {firstname: "Victor",
                   lastname:"Vianu"},
         title:  "Regular path queries...",
         page:   {first:122, last:133}}
}
```

Edge-labelled Graphs

Directed graph

- each node has a unique object identifier (oid)
- atomic values only on leaves
- oids are just strings or integers, so their meaning is restricted to a certain domain (would need URL or some such to locate objects across network)

SSD Syntax

$$\begin{aligned}
\langle ssd-expr\rangle &::= \langle value\rangle \mid \mathrm{oid}\langle value\rangle \mid \mathrm{oid}\\
\langle value\rangle &::= \mathrm{atomicvalue} \mid \langle complexvalue\rangle\\
\langle complexvalue\rangle &::= \{\mathrm{label} : \langle ssd-expr\rangle, \ldots \mathrm{label} : \langle ssd-expr\rangle\}
\end{aligned}$$

Consistent ssd-expression s

- any oid is defined at most once in s
- if an oid o is used in s, it must be defined in s.

In example above, we omitted oid's

Set semantics for ssd

Fact: relational databases can be represented as ssd
Variation of ssd model:

- same spirit as relational model, where a table is a set (no duplicates)
- wish `{a, a, b} = {a,b}`
- hence no oid's in this variation

Challenge: how to define equality of graphs (with cycles) ?

We'll use the concept of bisimulation. First define simulation.

Graph Simulation

Definition Let G_1 and G_2 be two edge-labelled graphs. A **simulation** is a relation R between the nodes such that

$$\forall (x_1, x_2) \in R . \forall (x_1, a, y_1) \in G_1 . \exists (x_2, a, y_2) \in G_2 . (y_1, y_2) \in R$$

Graph Bisimulation

Definition Let G_1 and G_2 be two edge-labelled graphs. A **bisimulation** is a relation R between the nodes such that both R and R^{-1} are simulations.

Set Semantics for SSD

Definition Two rooted graphs G_1 and G_2 are **equal** if there exists a bisimulation R from G_1 to G_2 such that $(\mathrm{root}(G_1), \mathrm{root}(G_2) \in R$.
Notation: $G_1 \sim G_2$

Examples.

Simulation vs. Bisimulation

Suppose R is a simulation between G_1 and G_2 and that S is a simulation between G_2 and G_1. Are G_2 and G_1 bisimilar ?

No

Facts about (Bi)Simulation

- The empty set is a (bi)simulation.
- If R and S are (bi)simulations, so is $R \cup S$
- Hence, there is a **maximal** (bi)simulation.
- To check whether $G_1 \sim G_2$: compute the maximal bisimulation R and test whether $(\mathrm{root}(G_1), \mathrm{root}(G_2) \in R$.

Computing a (Bi)Simulation

Computing the maximal (bi)simulation:

- Start with $R = \mathrm{nodes}(G_1) \times \mathrm{nodes}(G_2)$.
- While there exists $(x_1, x_2) \in R$ that violates the def'n, remove (x_1, x_2) from R.

This runs in polynomial time! Better algorithms exist:

- $O((m+n)\log(m+n))$ for bisimulation
- $O(mn)$ for simulation