University of Washington
Department of Computer Science & Engineering

Managing XML and Semistructured Data

Lecture 11: Schema - DTD's

Prof. Dan Suciu

Spring 2001

In this lecture

- Schema Extraction for SS data
- Schemas for XML
  - DTDs
  - XML Schema

Resources

- Extracting Schema from Semistructured Data Nestorov, Abiteboul, Motwani. SIGMOD 98
- **Data on the Web** Abiteboul, Buneman, Suciu : *section 3.3*

Review of Schemas so far

Upper bound schema S
- Tell us what labels are *allowed*
- Conformance test: D ≤ S
- In practice: need *deterministic* schemas

Lower bound schema S
- Tells us what labels are required
- Conformance test: S ≤ D
- Alternative formulation: datalog programs, maximal fixpoint

Schema Extraction (From Data)

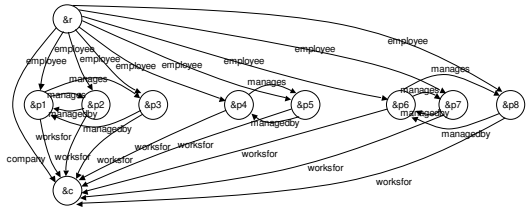
Problem statement
- given data instance D
- find the "most specific" schema S for D

In practice: S too large, need to relax

[Nestorov, Abiteboul, Motwani 1998]

Schema Extraction: Sample Data

Example database D =



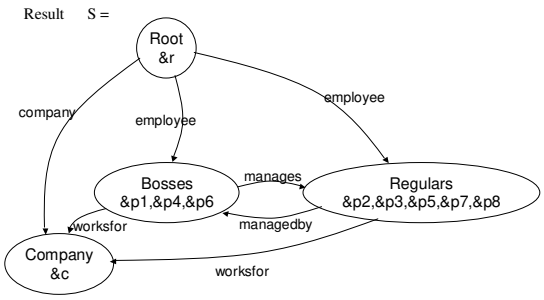
Lower Bound Schema Extraction

[NAM'98] approach:
- Start with the schema given by the data (S = D):
  - Each node = a predicate = a class

```
. . .
. . .
p4(x) :- link(x, manages, y), p5(y), link(x, worksfor, z), c(z)
p5(x) :- link(x, managed-by, y), p4(y), link(x, worksfor, z), c(z)
. . .
. . .
```

- Compute maximal fixpoint (PTIME)
- Declare two classes equal iff they are equal sets
  - E.g. p4={&p1,&p4,&p6}, p6={&p1,&p4,&p6}, hence p1=p4
- Equivalently, p=p' iff p(&p') and p'(&p)

Lower Bound Schema Extraction



Lower Bound Schema Extraction

Equivalently:

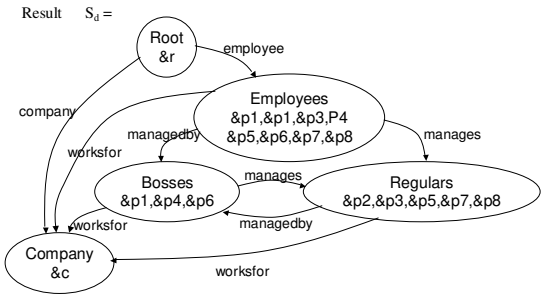
- Compute the maximal simulation $D \leq D$
  - Can do in time $O(m^2)$
- Two nodes p, p' are *equivalent* iff $x \leq x'$ and $x' \leq x$
- Schema consists of equivalence classes

Remark: could use the bisimulation relation instead (perhaps is even better)

Upper Bound Schema Extraction

- The extracted lower bound schema S is also an upper bound schema !
- But: nondeterministic
- Convert S ➔ S_d
- Alternatively, convert directly D ➔ $D_d = S_d$
  - These are *data guides* [McHugh and Widom]

Upper Bound Schema Extraction



XML Document Type Definitions

- part of the original XML specification
- an XML document may have a DTD
- terminology for XML:
  - well-formed: if tags are correctly closed
  - valid: if it has a DTD and conforms to it
- validation is useful in data exchange

Very Simple DTD

```
<!DOCTYPE company [
  <!ELEMENT company   ((person|product)*)>
  <!ELEMENT person  (ssn, name, office, phone?)>
  <!ELEMENT ssn       (#PCDATA)>
  <!ELEMENT name     (#PCDATA)>
  <!ELEMENT office    (#PCDATA)>
  <!ELEMENT phone    (#PCDATA)>
  <!ELEMENT product  (pid, name, description?)>
  <!ELEMENT pid    (#PCDATA)>
   <!ELEMENT description    (#PCDATA)>
]>
```

Very Simple DTD

Example of valid XML document:

```
<company>
   <person>  <ssn> 123456789 </ssn>
             <name> John </name>
             <office> B432 </office>
             <phone> 1234 </phone>
   </person>
   <person>  <ssn> 987654321 </ssn>
             <name> Jim </name>
             <office> B123 </office>
   </person>
   <product> ...  </product>
   ...
</company>
```

Content Model

- Element *content*: what we can put in an element (aka *content model*)
- Content model:
  - Complex = a regular expression over other elements
  - Text-only = #PCDATA
  - Empty = EMPTY
  - Any = ANY
  - Mixed content = (#PCDATA | A | B | C)*
    - (i.e. very restrictied)

Attributes in DTDs

```
<!ELEMENT person  (ssn, name, office, phone?)>
<!ATTLIS     person age CDATA  #REQUIRED>
```

```
<person age="25">
     <name> ....</name>
      ...
</person>
```

Attributes in DTDs

```
<!ELEMENT person  (ssn, name, office, phone?)>
<!ATTLIS     person age          CDATA  #REQUIRED
                    id           ID     #REQUIRED
                    manager      IDREF  #REQUIRED
                    manages      IDREFS #REQUIRED
>
```

```
<person age="25"
        id="p29432"
        manager="p48293"  manages="p34982 p423234">
    <name> ....</name>
     ...
</person>
```

Attributes in DTDs

Types:

- CDATA = string
- ID = key
- IDREF = foreign key
- IDREFS = foreign keys separated by space
- (Monday | Wednesday | Friday) = enumeration
- NMTOKEN = must be a valid XML name
- NMTOKENS = multiple valid XML names
- ENTITY = you don't want to know this

Attributes in DTDs

Kind:

- #REQUIRED
- #IMPLIED = optional
- value = default value
- value #FIXED = the only value allowed

Using DTDs

- Must include in the XML document
- Either include the entire DTD:
  - <!DOCTYPE rootElement [.....]>
- Or include a reference to it:
  - <!DOCTYPE rootElement SYSTEM "http://www.mydtd.org">
- Or mix the two... (e.g. to override the external definition)

DTDs as Grammars

```
<!DOCTYPE paper [
  <!ELEMENT paper   (section*)>
  <!ELEMENT section ((title,section*) | text)>
  <!ELEMENT title     (#PCDATA)>
  <!ELEMENT text      (#PCDATA)>
]>
```

```
<paper> <section> <text> </text> </section>
        <section> <title> </title> <section> … </section>
                                   <section> … </section>
        </section>
</paper>
```

DTDs as Grammars

- A DTD = a grammar
- A valid XML document = a parse tree for that grammar

DTDs as Schemas

Not so well suited:

- impose unwanted constraints on order
  <!ELEMENT person (name,phone)>
- references cannot be constrained
- can be too vague:
  <!ELEMENT person ((name|phone|email)*)>

like an upper bound schema