

Data Integration: XML

Jan Chomicki

University at Buffalo

XML documents (simplified)

XML tree

- finite, ordered, unranked tree
- element, attribute and text nodes
- element and attribute node labels from a finite label alphabet Σ ($label^t$)
- attribute and text (`#PCDATA`) values from an infinite domain D
- only element nodes have children
- document order (left-to-right prefix order)

XML trees represent **well-formed documents**:

- matching, properly nested opening and closing tags
- single root element

Regular expressions over Σ

$$E := \varepsilon \mid a \mid E \cup E \mid E E \mid E^* \text{ where } a \in \Sigma.$$

Defining valid XML documents

XML schema definitions

- Document Type Definitions (DTDs)
- XML Schema
- Nondeterministic tree automata

DTD (over Σ)

- **element-only** content: a function mapping each node label from Σ to a regular expression to which the concatenated children of the node must conform
- regular expression is **1-unambiguous**
 - parse with a single-symbol lookahead
 - counterexample: $(a + b)^*aa^*$ and baa .
- also text-only, mixed, empty, and unrestricted content
- **attributes**: text-valued (CDATA), enumerations, ID, IDREF

XML Schema

Simple types

- **base** types (many)
- **derived** types (by constraining facets)
- **list/union** types

Complex types

- **content model**: sequence, all, choice
- attribute declarations
- types can be recursive or anonymous
- element types can be locally declared

Integrity constraints

- keys
- foreign keys

Tree automata

Nondeterministic tree automaton (NTA)

A tuple $B = (Q, \Sigma, \delta, F)$:

- Q : a finite set of states
- $F \subseteq Q$: the set of final states
- $\delta : Q \times \Sigma \rightarrow 2^{Q^*}$, where Q^* is the set of finite words over Q and $\delta(q, a)$ is a regular string language over Q for every $q \in Q$ and $a \in \Sigma$.

Run of B over a tree t

Labeling $\lambda : Dom(t) \rightarrow Q$ such that for every $v \in Dom(t)$ with n children v_1, \dots, v_n , $\lambda(v_1) \cdots \lambda(v_n) \in \delta(\lambda(v), label^t(v))$. A run is **accepting** if $\lambda(root) \in F$.

Expressive power

DTDs and XML Schema (w/o attributes and text) describe tree languages recognized by NTAs.

XPath

Data model

- tree-based
- nodes: root, element, attribute, text,...
- document order: left-to-right prefix traversal

Path expression

- describes a set of paths in a document
- returns a sequence of nodes in document order
- evaluated in a *context*:
 - node
 - position
 - size
- absolute (starting at root) or relative
- consists of steps separated by / or //
- wildcards
- union (|), intersection, difference

axis::nodeTest stepQualifiers

- *axis*:
 - *forward*: child, descendant, following-sibling, following, self, descendant-or-self
 - *backward*: parent, ancestor, preceding-sibling, preceding, ancestor-or-self
 - attribute
- *node test*: name test (name or wildcard), kind test
- *step qualifiers*: predicate expressions (in square brackets)

Abbreviated syntax

- ① child is the default axis, can be omitted
- ② the attribute axis can be abbreviated to @
- ③ // is short for /descendant-or-self::node()/
- ④ . is short for self::node()
- ⑤ .. is short for parent::node()
- ⑥ a positive integer K is short for [position()=K]

Integrity constraints in XML Schema

Keys

```
<( key | unique) name=KeyName>  
<selector xpath=Path>  
<field xpath=Path1>  
...  
<field xpath=PathN>  
</key>
```

Foreign keys

```
<keyref name=RefName refer=KeyName>  
<selector xpath=Path>  
<field xpath=Path1>  
...  
<field xpath=PathN>  
</keyref>
```

Features

- functional
- compositional: expressions can be nested arbitrarily
- recursion
- declarative: influenced by SQL (and OQL)

XQuery expressions

- Constants: numbers, strings,...
- Variables
- XPath expressions
- Element/attribute constructors
- Operators and functions: arithmetic,...
- FLWOR expressions
- Quantifiers
- Aggregation
- User-defined functions

FLWOR expressions

```
for variableRangeSpecifications  
let variableDefinitions  
where condition  
order by orderExpression  
return resultExpression
```

User-defined functions

```
declare function Name(Arguments)  
as Type  
{Expression}
```

Storing XML documents in relational databases

Storing nodes and edges of the document tree

- a binary edge relation
- implementing XPath requires recursion (SQL3)

Encoding the tree structure using ranges

- range of child \subset range of parent
- queries w/o recursive functions can be translated to SQL2