NC STATE UNIVERSITY

ENGINEERING ONLINE

Lecture Notes

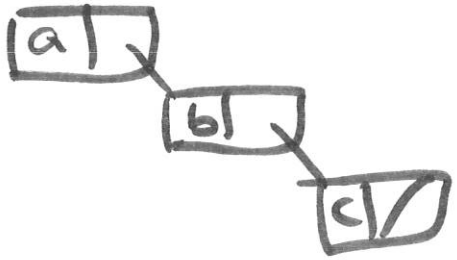
Course Number: CSC 513

Instructor: Dr. Singh

Lecture Number: 21

Compare with Lisp

(a b c)

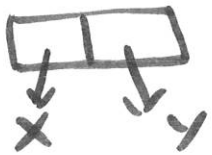


List processing language

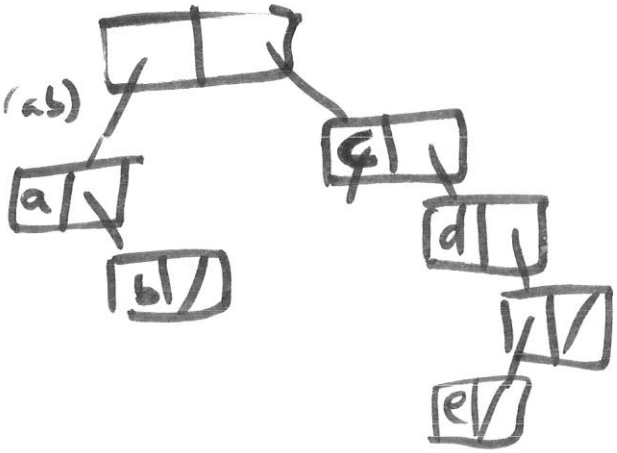
BINARY TREES

- S-expressions
- Cons pairs: **car** and **cdr**
- Lists as nil-terminated s-expressions
- Arbitrary structures built from few primitives
- Untyped
- Easy parsing :
- Regularity of structure encourages recursion

(X . Y)



((a b) (c d (e)))



(a b
(a b))
(a)) b (
(a b) (c d)

edge (origin A) origin edge

dest (dest B) dest

edge

<edge>
<origin> A </origin>
<dest> B </dest>
</edge>

 abc
(b (i abc) b) i

} not well formed XML
→ no tree

((abc))

(bold (italic abc))

} Lisp relies on programmer

Meaning in XML

- Relational DBMSs work for highly structured information, but rely on column names for meaning
- Same problem in XML (reliance on names for meaning) but better connections to richer meaning representations

Leads to need for a richer way of specifying such names

i.e. a vocabulary

XML Namespaces: 1

relies on classpath { package =

import a.List;
import java.util.List;
... extends implements a.List
a.List x = ...

- Because XML supports custom vocabularies and interoperation, there is a high risk of name collision
- A namespace is a collection of names
- Namespaces must be identical or disjoint
  - Crucial to support independent development of vocabularies
  - (3) MAC addresses
  - (1) Postal and (2) telephone codes
  - (4) Vehicle identification numbers
  - Domains as for the Internet
  - On the Web, use URIs for uniqueness

(defmacro

IP address

Rely on a NAMING convention

Which of the above naming conventions can we RESOLVE? (to a resource)

Uniform Resource Identifier

- URIs are abstract
- What matters is their (purported) uniqueness
- URIs have no proper syntax per se
- Kinds of URIs
  - URLs, as in browsing: not used in standards any more
  - URNs, which leave the mapping of names to locations up in the air
- Good design: the URI resource exists
  - Ideally, as a description of the resource in RDDL
  - Use a URL or URN

(1) particular syntax
(2) way to resolve

XML Namespaces: 2

qualified name

ns : name

```
<!-- xml* is reserved -->
<?xml version="1.0"?>
<arbit:top xmlns="a URI" <!-- default namespace -->
  xmlns:arbit="http://wherever.it.might.be/arbit-ns"
  xmlns:random="http://another.one/random-ns">
 <arbit:aElem attr1="v1" attr2="v2">
   Optional text (PCDATA)
   <arbit:bElem attr1="v1" attr2="v2"/>
 </arbit:aElem>
 <random:simple_elem/>
 <random:aElem attr3="v3"/>
  <!-- compare arbit:aElem -->
</arbit:top>
```

RDDL

Resource Directory Description Language

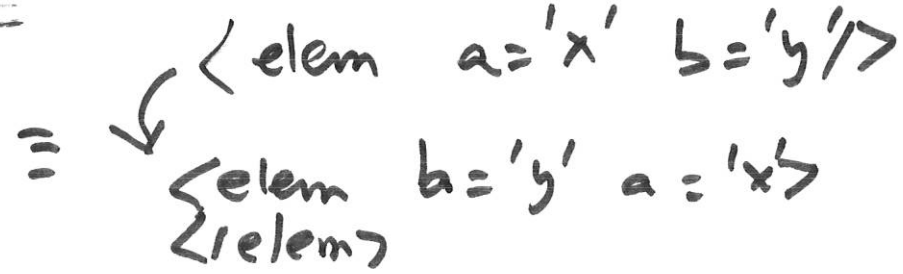
- Meant to solve the problem that a URI may not have any real content, but people expect to see some (human readable) content
- Captures namespace description for people
  - XML Schema
  - Text description

Well-Formedness and Parsing

Doc → Tree

- An XML document maps to a parse tree (if well-formed; otherwise not XML)
  - Each element must end (exactly *once*): obvious nesting structure (one root)
  - An attribute can have at most one occurrence within an element; an attribute's value must be a quoted string
- Well-formed XML documents can be parsed

XML InfoSet

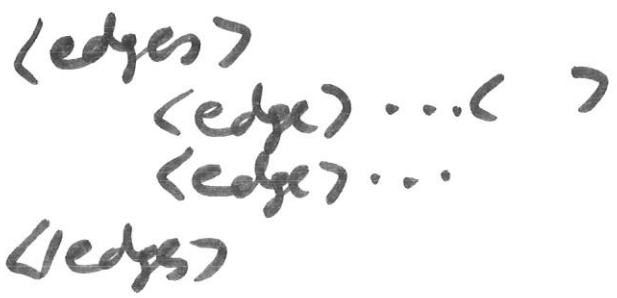


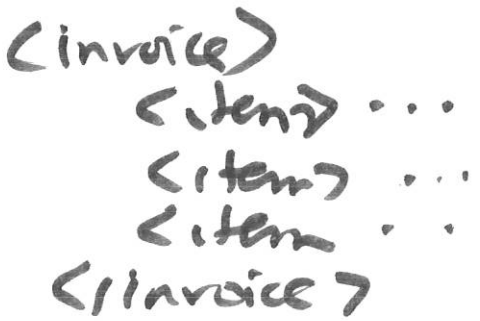
A standardization of the low-level aspects of XML

- What an element looks like
- What an attribute looks like
- What comments and namespace references look like
- Ordering of attributes is irrelevant
- Representations of strings and characters

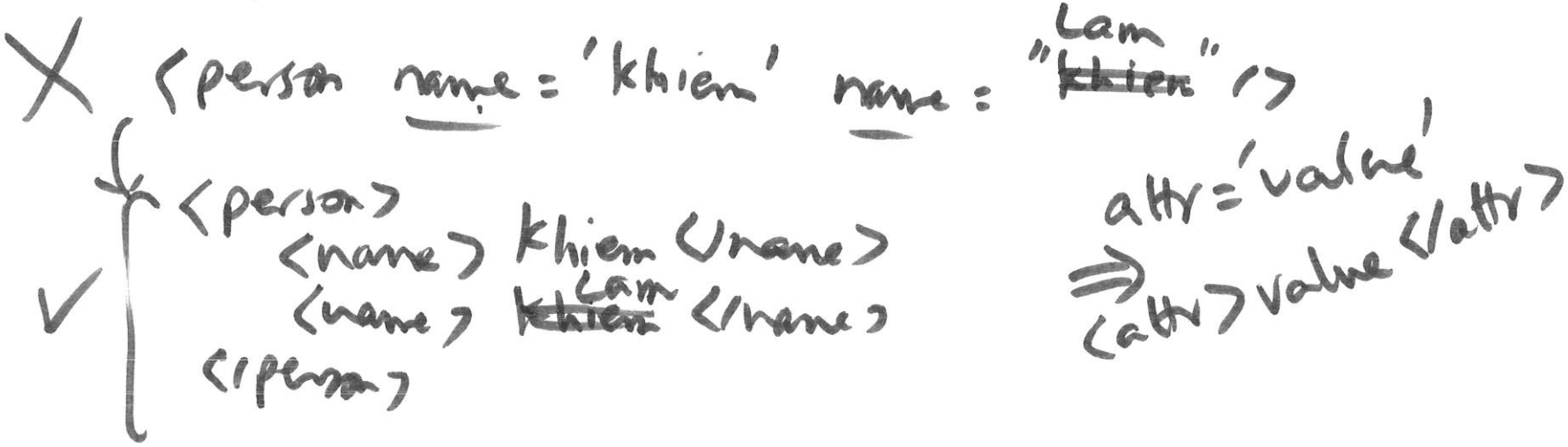
Primarily directed at tool vendors

Elements Versus Attributes: 1





- Elements are essential for XML (tree): structure and expressiveness
  - Have subelements and attributes
  - Can be repeated
  - Loosely might correspond to independently existing entities
  - Can capture all there is to attributes



Elements Versus Attributes: 2

- Attributes are not essential
  - End of the road: no subelements or attributes
  - Like text; restricted to string values
  - Guaranteed unique for each element
  - Capture adjunct information about an element
  - Great as references to elements

Good idea to use in such cases to improve readability

Elements Versus Attributes: 3

```
<invoice>
 <price currency='USD'>
  19.95
 </price>
</invoice>
```

Or

```
<invoice amount='19.95' currency='USD'/>
```

Or even

```
<invoice amount='USD 19.95'/>
```

RELIES on parsing a string to extract the essential structure

'invoice, USD, 19.95'

Outline

XML Query and Manipulation

Main XML query and manipulation languages include

- ✔ XPath
- ✔ XQuery
- XSLT
- ✔ SQL/XML

Metaphors for Handling XML: 1

How we conceptualize XML documents determines our approach for handling them

invoice, price, currency, USD
19.95, price invoice

- *Text:* an XML document is text
  - Ignore any structure and perform simple pattern matches
- *Tags:* an XML document is text interspersed with tags
  - Treat each tag as an "event" during reading a document, as in SAX (Simple API for XML)
  - Construct regular expressions as in screen scraping

ABOLISH THE WORD 'TAG' FROM YOUR VOCABULARY

Metaphors for Handling XML: 2

- *Tree:* an XML document is a tree
  - Walk the tree using DOM (Document Object Model)
- *Template:* an XML document has regular structure
  - Let XPath, XSLT, XQuery do the work
- *Thought:* an XML document represents an information model
  - Access knowledge via RDF or OWL