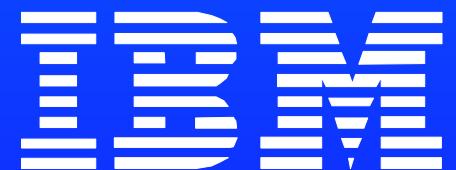




Efficient encoding of XML updates

Terrence Poon
Francisco Curbera
David A. Epstein

IBM Research Division
T.J. Watson Research Center
August 20th, 1999

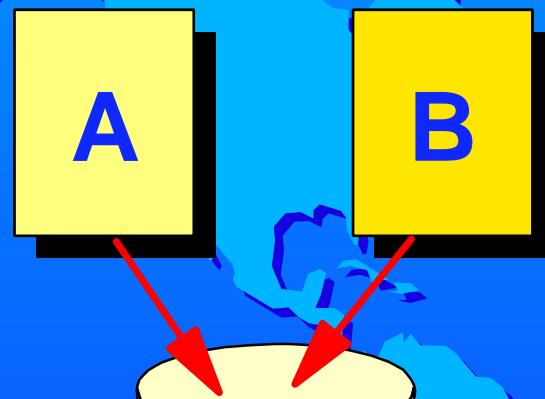


Overview

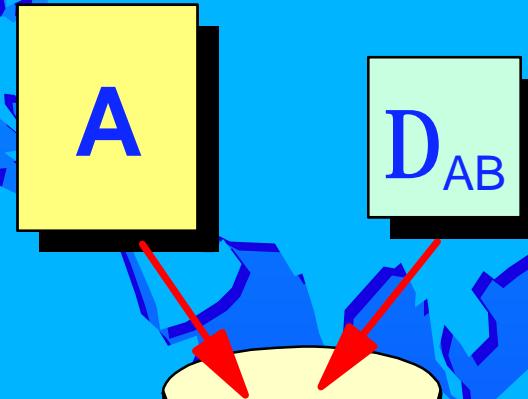
- Introduction
- Motivation
- XML Update Language (XUL)
- Patch algorithm
- Applications / demo
- Future work: adding semantics to XUL

Introduction

Differentiation



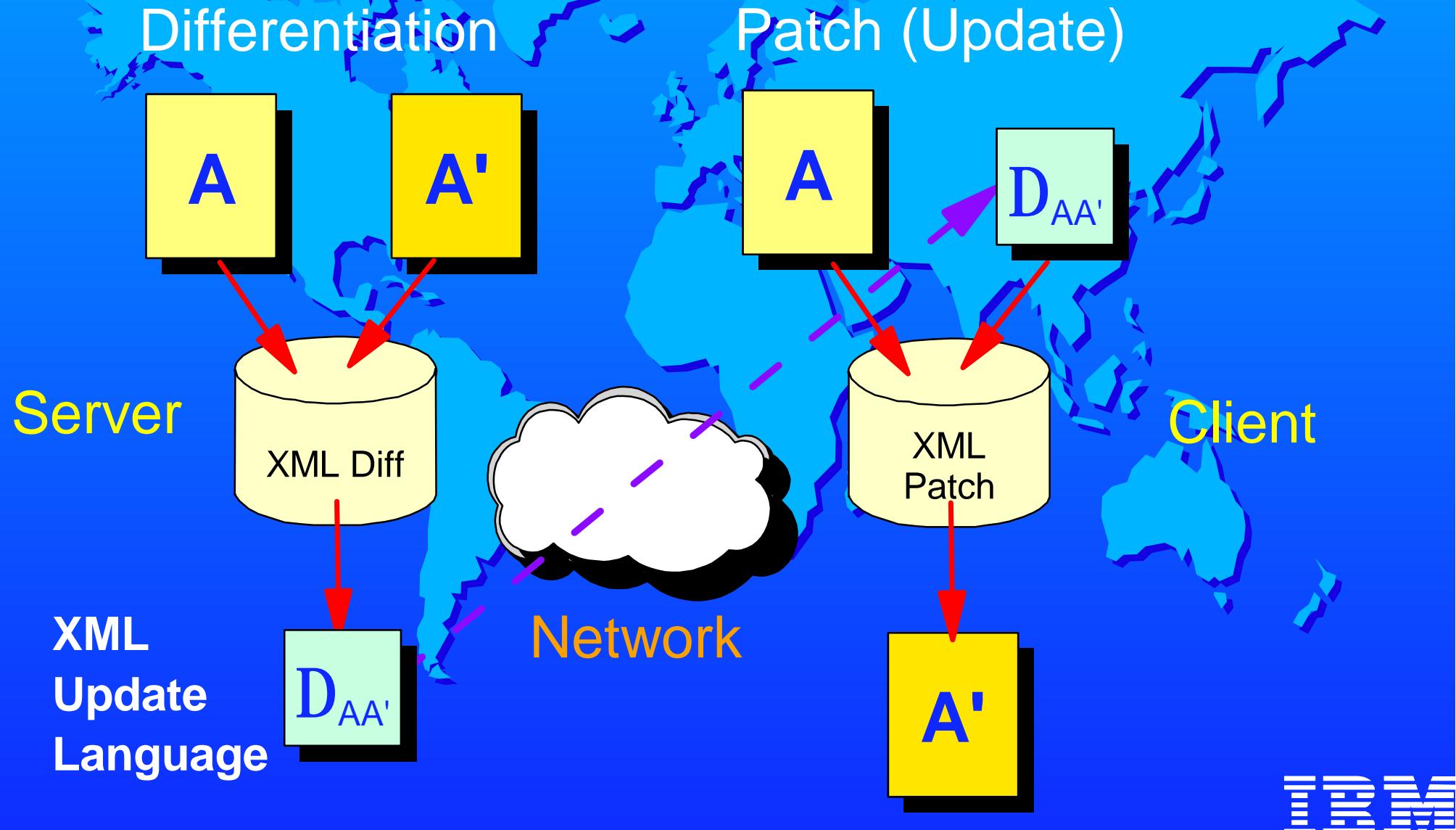
Patch (Update)



XML
Update
Language



Sample Scenario



Motivation

- Small update vs. large document
- Advantages over "flat" diff and patch
 - Understands the tree structure of an XML document
 - Avoids problems from multiple surface string representations
 - Operates directly on DOM - no re-parsing needed

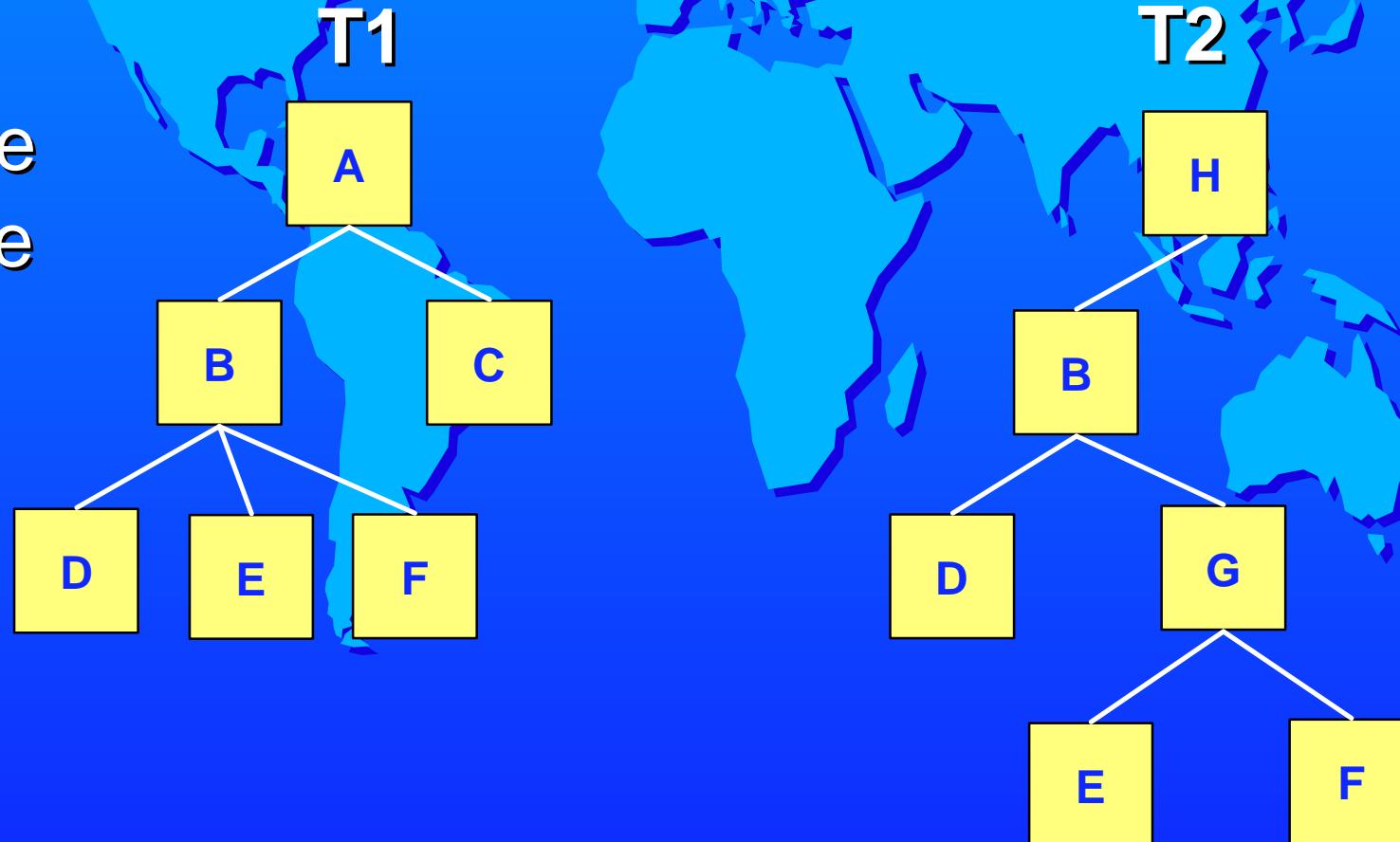
Tree-to-tree correction problem

- Find the least expensive sequence of operations to transform T1 to T2

add

remove

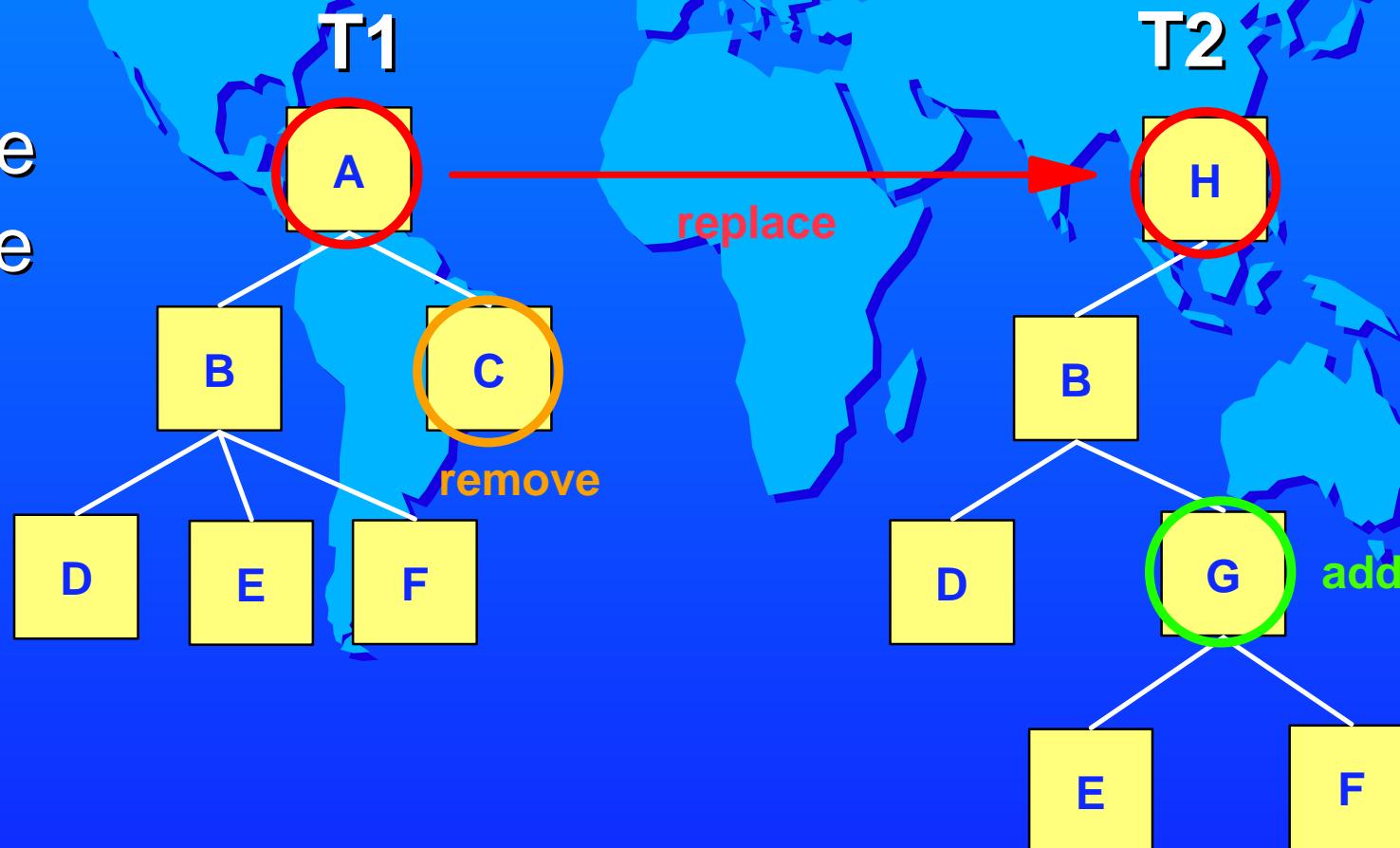
replace



Tree-to-tree correction problem

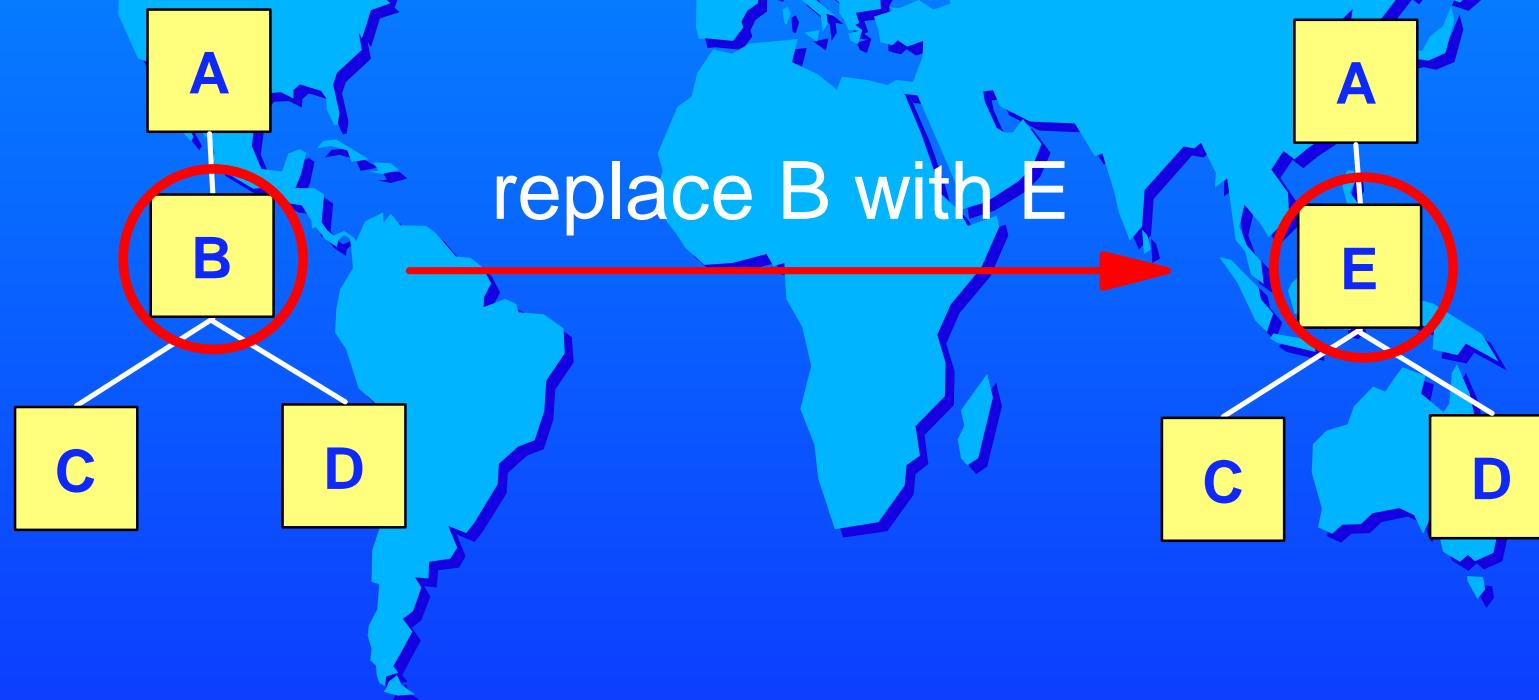
- Find the least expensive sequence of operations to transform T1 to T2

add
remove
replace



Basic operations

- Replace



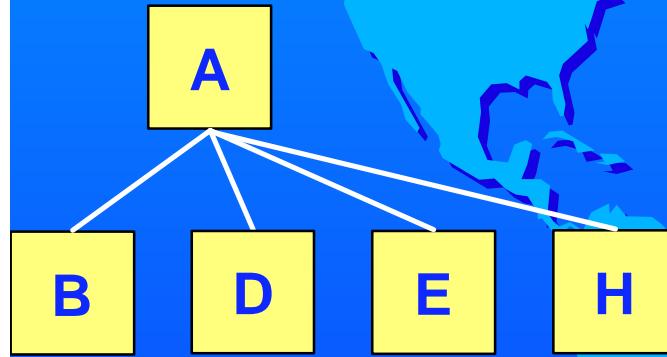
Basic operations

- Remove



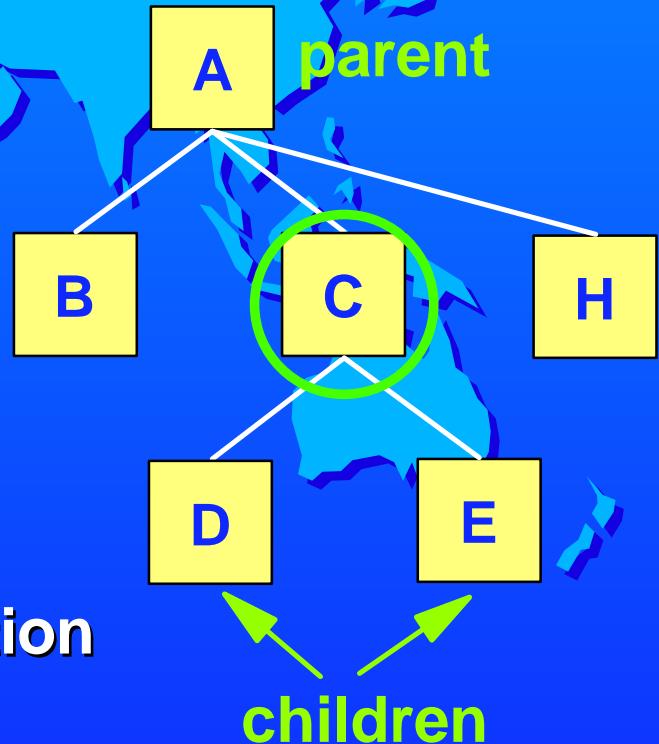
Basic operations

■ Add



add C,
with A as parent,
B as psib, and
D and E as children

previous
sibling



- need to specify additional information
- inverse of remove

Encoding document updates

- Language requirements:
 - XML-based
 - Basic operations: add, remove, replace
 - Aggregate operations: prune, graft, move
 - Expresses dependencies between operations
 - Refers only to initial and final states of document
- Naive approach (FUL)
- XML Update Language (XUL)



Naive approach (FUL)

- Flat list of operations
- like Unix 'diff' output

```
<diff>
  <add match="xfl1" type="1" name="b" parent="xfl0"/>
  <remove match="/*[1]/*[2]/*[1]" />
  <add match="xfl2" type="1" name="c" parent="xfl0"
    psib="xfl1"/>
  <add match="xfl0" type="1" name="a"
    parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />
</diff>
```



Naive approach (FUL)

- Flat list of operations
- like Unix 'diff' output

The diagram features a world map background. Three arrows point from text labels to specific parts of the XML code:

- A green arrow points from the word "operations" to the opening tag <diff>.
- An orange arrow points from the word "references" to the attribute "match" in the first <add> element.
- A yellow arrow points from the phrase "(symbolic / XPath)" to the attribute "psib" in the third <add> element.

```
<diff>
  <add match="xfl1" type="1" name="b" parent="xfl0"/>
  <remove match="/*[1]/*[2]/*[1]" />
  <add match="xfl2" type="1" name="c" parent="xfl0"
    psib="xfl1"/>
  <add match="xfl0" type="1" name="a"
    parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />
</diff>
```

Naive approach (FUL)

- Flat list of operations
- like Unix 'diff' output

```
<diff>
  <add match="xf11" type="1" name="b"
    parent="xf10"/>
  <remove match="/*[1]/*[2]/*[1]" />
  <add match="xf12" type="1" name="c"
    parent="xf10" psib="xf11"/>
  <add match="xf10" type="1" name="a"
    parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />
</diff>
```

The world map background features several yellow arrows originating from the XML code above and pointing to specific countries. One arrow points from the first 'add' element to Australia. Another points from the 'remove' element to the United States. A third points from the second 'add' element to Canada. A fourth points from the third 'add' element to Mexico. A fifth points from the same 'add' element to Chile. A sixth points from the same 'add' element to Argentina. A seventh points from the same 'add' element to Brazil. A eighth points from the same 'add' element to South Africa. A ninth points from the same 'add' element to Japan. A tenth points from the same 'add' element to India.

Problems: naive approach (FUL)

- Abundance of symbolic references
 - ex. xfl0, xfl1, etc.
 - due to structured nature of XML

```
<diff>
```

```
  <add match="xfl1" type="1" name="b" parent="xfl0"/>
```

```
  <remove match="/*[1]/*[2]/*[1]" />
```

```
  <add match="xfl2" type="1" name="c" parent="xfl0"  
    psib="xfl1"/>
```

```
  <add match="xfl0" type="1" name="a"  
    parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />
```

```
</diff>
```



Problems: naive approach (FUL)

- Implicit dependencies
 - need topological sorting

```
<diff>
  <add match="xfl1" type="1" name="b" parent="xfl0"/>
  <remove match="/*[1]/*[2]/*[1]" />
  <add match="xfl2" type="1" name="c" parent="xfl0"
    psib="xfl1"/>
  <add match="xfl0" type="1" name="a"
    parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />
</diff>
```



Problems: naive approach (FUL)

- Implicit dependencies
 - need topological sorting
 - difficult to parallelize (need partitioning)

<diff>

<remove match="/*[1]/*[2]/*[1]" />

<add match="xfl0" type="1" name="a"

parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />

<add match="xfl1" type="1" name="b" parent="xfl0"/>

<add match="xfl2" type="1" name="c" parent="xfl0"

psib="xfl1" />

</diff>



Problems: naive approach (FUL)

- Hard to interpret
 - unstructured
 - absolute paths - no context

```
<diff>
  <add match="xfl1" type="1" name="b" parent="xfl0"/>
  <remove match="/*[1]/*[2]/*[1]" />
  <add match="xfl2" type="1" name="c" parent="xfl0"
    psib="xfl1"/>
  <add match="xfl0" type="1" name="a"
    parent="/*[1]/*[2]" psib="/*[1]/*[2]/*[2]" />
</diff>
```



XML Update Language (XUL)

- structured, contextual language

```
<node id="/*[1]/*[2]">
  <node id="./*[1]" op="remove"/>
  <node id="./*[2]" />
  <node op="add" type="1" name="a">
    <node op="add" type="1" name="b"/>
    <node op="add" type="1" name="c"/>
  </node>
</node>
```



XML Update Language (XUL)

- structured, contextual language

```
<node id="/*[1]/*[2]">
  <node id="./*[1]" op="remove"/>
  <node id="./*[2]" />
  <node op="add" type="1" name="a">
    <node op="add" type="1" name="b"/>
    <node op="add" type="1" name="c"/>
  </node>
</node>
```

node references

operations

The diagram illustrates the structure of an XML Update Language (XUL) document. It features a world map background. Overlaid on the map are two types of annotations: yellow arrows pointing to specific nodes in the XML code, labeled 'node references', and green arrows pointing to the 'op' attribute values ('remove', 'add'), labeled 'operations'. The XML code itself is displayed in white text against the blue/purple background, showing how nodes are selected and modified.

XUL: Simple Example

```
<node id="/*[1]">  
  <node id="/*[2]" />  
  <node op="add" name="d"  
    type="1" />  
</node>
```

Update

The diagram illustrates the flow of an XUL update. It features a world map background. On the left, under the heading 'Update', is the XUL code. Two yellow arrows point from the first two nodes to the corresponding elements in the 'Source' section. A dashed orange arrow points from the 'Result' section back down to the 'Source' section, indicating the propagation of changes.

```
<a>  
<b/>  
<c/>  
</a>
```

Source

```
<a>  
<b/>  
<c/>  
<d/>  
</a>
```

Result



XML Update Language

- structured, contextual language
 - contextual references

```
<node id="/*[1]/*[2]">
  <node id="./*[1]" op="remove"/>
  <node id="./*[2]" />
  <node op="add" type="1" name="a">
    <node op="add" type="1" name="b"/>
    <node op="add" type="1" name="c"/>
  </node>
</node>
```



XML Update Language

- structured, contextual language
 - explicit dependencies
 - easily parallelizable

```
<node id="/*[1]/*[2]">
  <node id="./*[1]" op="remove"/>
  <node id="./*[2]">
    <node op="add" type="1" name="a">
      <node op="add" type="1" name="b"/>
      <node op="add" type="1" name="c"/>
    </node>
  </node>
</node>
```



XML Update Language

- structured, contextual language
 - easier to interpret
 - reproduces meaningful parts of the source
 - uses relative paths

```
<node id="/*[1]/*[2]">
  <node id="./*[1]" op="remove"/>
  <node id="./*[2]" />
  <node op="add" type="1" name="a">
    <node op="add" type="1" name="b"/>
    <node op="add" type="1" name="c"/>
  </node>
</node>
```



XML Update Language

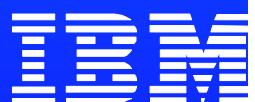
- faster to patch than FUL (~15%)
 - no need for topological sorting
 - uses relative paths

```
<node id="/*[1]/*[2]">
  <node id="./*[1]" op="remove"/>
  <node id="./*[2]">
    <node op="add" type="1" name="a">
      <node op="add" type="1" name="b"/>
      <node op="add" type="1" name="c"/>
    </node>
  </node>
</node>
```



Patch algorithm

- Input: source document and update document
- Output: updated version of the document
- Two pass algorithm
 - 1. Resolve the node references.
 - 2. Perform the operations using a depth-first traversal of the XUL.
- Possibly inconsistent or invalid intermediate states



Resolving node references (XPath)

```
<n id="/*[1]/*[2]">
  <n id="./*[1]" op="r"/>
  <n id="./*[2]"/>
  <n op="a" type="1" name="a">
    <n op="a" type="1" name="b"/>
    <n op="a" type="1" name="c"/>
  </n>
</n>
```

```
<?xml version="1.0"?>
<e>
  <f/>
  <g>
    <h/>
    <i/>
  </g>
</e>
```

Update

Source



Performing the operations (remove)

```
<n id="/*[1]/*[2]">
  <n id="./*[1]" op="r"/>
  <n id="./*[2]"/>
  <n op="a" type="1" name="a">
    <n op="a" type="1" name="b"/>
    <n op="a" type="1" name="c"/>
  </n>
</n>
```

```
<?xml version="1.0"?>
<e>
  <f/>
  <g>
    <h/>
    <i/>
  </g>
</e>
```

Update

Source



Perform the operations (add)

```
<n id="/*[1]/*[2]">  
  <n id="./*[1]" op="r"/>  
  <n id="./*[2]" />  
    <n op="a" type="1" name="a">  
      <n op="a" type="1" name="b"/>  
      <n op="a" type="1" name="c"/>  
    </n>  
</n>
```

Update

```
<?xml version="1.0"?>  
<e>  
  <f/>  
  <g>  
    <i/>  
  </g>  
</e>
```

Intermediate state



Performing the operations (add)

```
<n id="/*[1]/*[2]">  
  <n id="./*[1]" op="r"/>  
  <n id="./*[2]" />  
    <n op="a" type="1" name="a">  
      <n op="a" type="1" name="b"/>  
      <n op="a" type="1" name="c"/>  
    </n>  
</n>
```

```
<?xml version="1.0"?>  
<e>  
  <f/>  
  <g>  
    <i/>  
    <a/>  
  </g>  
</e>
```

Update

Intermediate state



Performing the operations (add)

```
<n id="/*[1]/*[2]">  
  <n id="./*[1]" op="r"/>  
  <n id="./*[2]" />  
    <n op="a" type="1" name="a">  
      <n op="a" type="1" name="b"/>  
      <n op="a" type="1" name="c"/>  
    </n>  
</n>
```

```
<?xml version="1.0"?>  
<e>  
  <f/>  
  <g>  
    <i/>  
    <a>  
      <b/>  
    </a>  
  </g>  
</e>
```

Update

Intermediate state



Performing the operations (add)

```
<n id="/*[1]/*[2]">  
  <n id="./*[1]" op="r"/>  
  <n id="./*[2]" />  
    <n op="a" type="1" name="a">  
      <n op="a" type="1" name="b"/>  
      <n op="a" type="1" name="c"/>  
    </n>  
</n>
```

```
<?xml version="1.0"?>  
<e>  
  <f/>  
  <g>  
    <i/>  
    <a>  
      <b/>  
      <c/>  
    </a>  
    </g>  
</e>
```

Update

Updated document



DOM-related issues:

Inconsistent intermediate states

- The document root can only have one element child.
 - Intermediate state: the document root may have multiple children

Source

```
<a>  
<b/>  
<c/>  
</a>
```

Update (XUL)

```
<node id="/">  
<node id="./*[1]" op="remove">  
  <node id="./*[1]" op="remove"/>  
  </node>  
</node>
```



DOM-related issues:

Inconsistent intermediate states

- The document root can only have one element child.
 - Intermediate state: the document root may have multiple children

Source

```
<b/>  
<c/>
```

Illegal!

Update (XUL)

```
<node id="/">  
  <node id="./*[1]" op="remove">  
    <node id="./*[1]" op="remove"/>  
  </node>  
</node>
```



DOM-related issues:

Inconsistent intermediate states

- The document root can only have one element child.
 - Intermediate state: the document root may have multiple children
 - Solution: insert a placeholder root

Source

```
<placeholder>  
  <a><!-->  
  <b><!-->  
  <c/>  
  </a>  
</placeholder>
```

Update (XUL)

```
<node id="/">  
  <node id="./*[1]" op="remove">  
    <node id="./*[1]" op="remove"/>  
    </node>  
  </node>
```



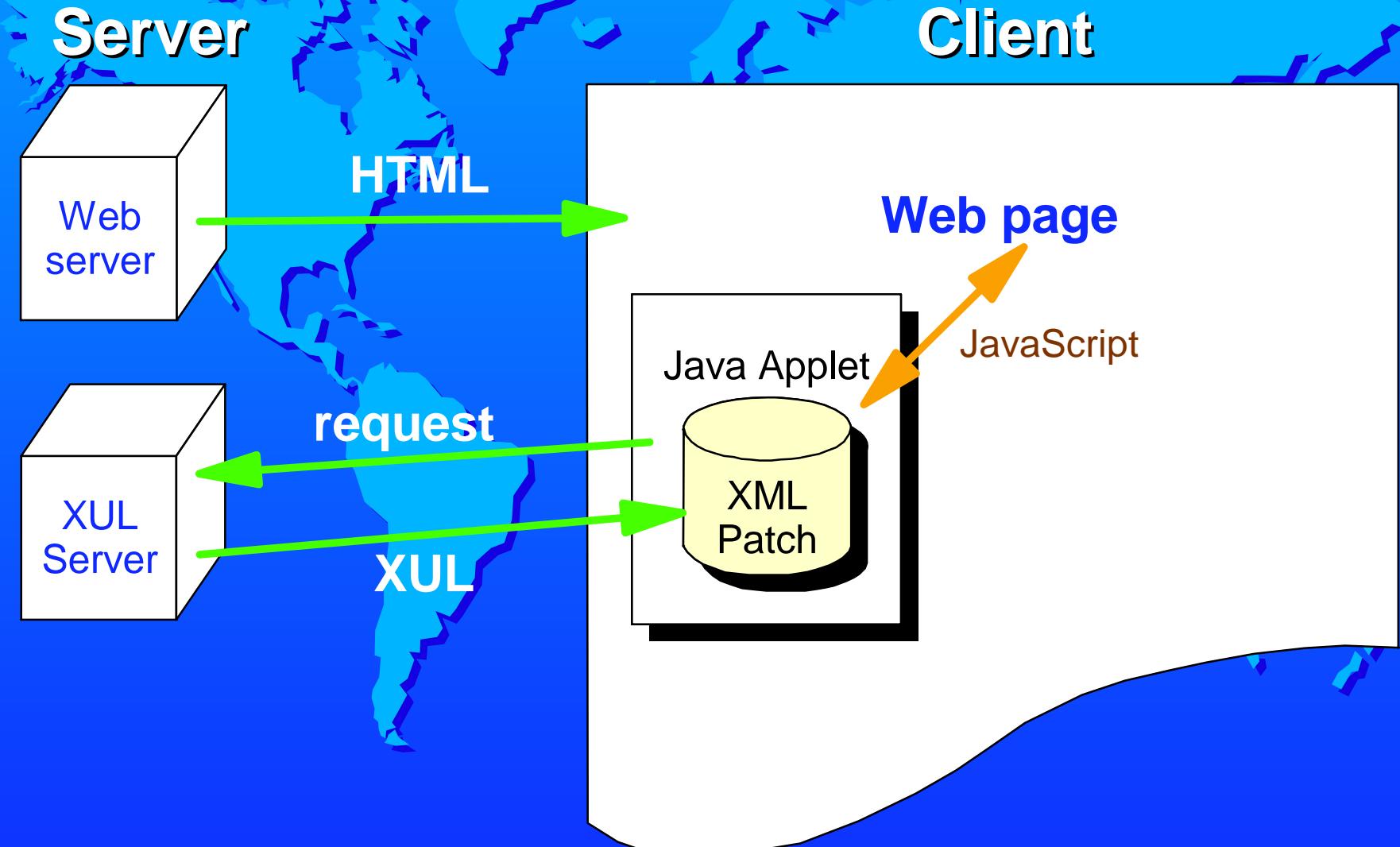
Applications

- Update from a central server
 - i.e. web pages
- Version control (like RCS / CVS)
- Collaborative editing of a document
- Selective update using an interactive editor
- Mirroring of large archives

The point

■ Bandwidth reduction!

Demo (IBM WebSphere FAQ)



IBM

Adding semantics to XUL

- Current XUL: structural description
- Future work: relate XUL to semantic framework of the document

References

- H. Maruyama, K. Tamura, and R. Uramoto, "Digest Values for DOM (DOMHash) Proposal,"
<http://w3e.trl.ibm.com/projects/xml/domhash.htm>, June 1998.
- World Wide Web Consortium, "Document Object Model (DOM) Level 1 Specification,"
<http://www.w3.org/TR/REC-DOM-Level-1/>, October 1998.
- World Wide Web Consortium, "XML Path Language (XPath)," <http://www.w3.org/TR/xpath>, July 1999.
- K. Zhang and D. Sasha, "Simple Fast Algorithm for the Editing Distance between Trees and Related Problems,"
SIAM J. Comput., 18, (6), 1245-1262, (1989).



XML TreeDiff

- **IBM alphaWorks**

<http://www.alphaworks.ibm.com/formula/xmltreediff>

- **Contacts:**

- David Epstein

epstein@us.ibm.com

- Francisco Curbera

curbera@us.ibm.com

