Large XML on Small Devices: Techniques Developed in the Fuego Core Project

Helsinki-Rutgers Ph.D. Workshop 2007

Tancred Lindholm, Jaakko Kangasharju

{tancred.lindholm,jkangash}@hiit.fi

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

XML Pros and Cons

- XML is
  - text-based
  - free-form (not fixed-size records)
  - verbose (descriptive tag names, whitespace)
- These properties decrease performance viz. binary formats
  - parsing/serialization needed
  - marshalling needed
  - more storage needed

HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

Why XML on Mobile Phones?

- Binary formats seem to be the right thing to do on constrained devices
- However, XML on the phone **keeps things simple**
  - avoid data transcoding when interchanging data
  - leverage XML ecosystem
  - don't force new formats on developers
  - facilitate debugging
- Mobile phones nowadays support (small) XML
- Phone storage capacity has increased rapidly
  - Several GB is not uncommon
  - XML verbosity becomes less of a problem

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

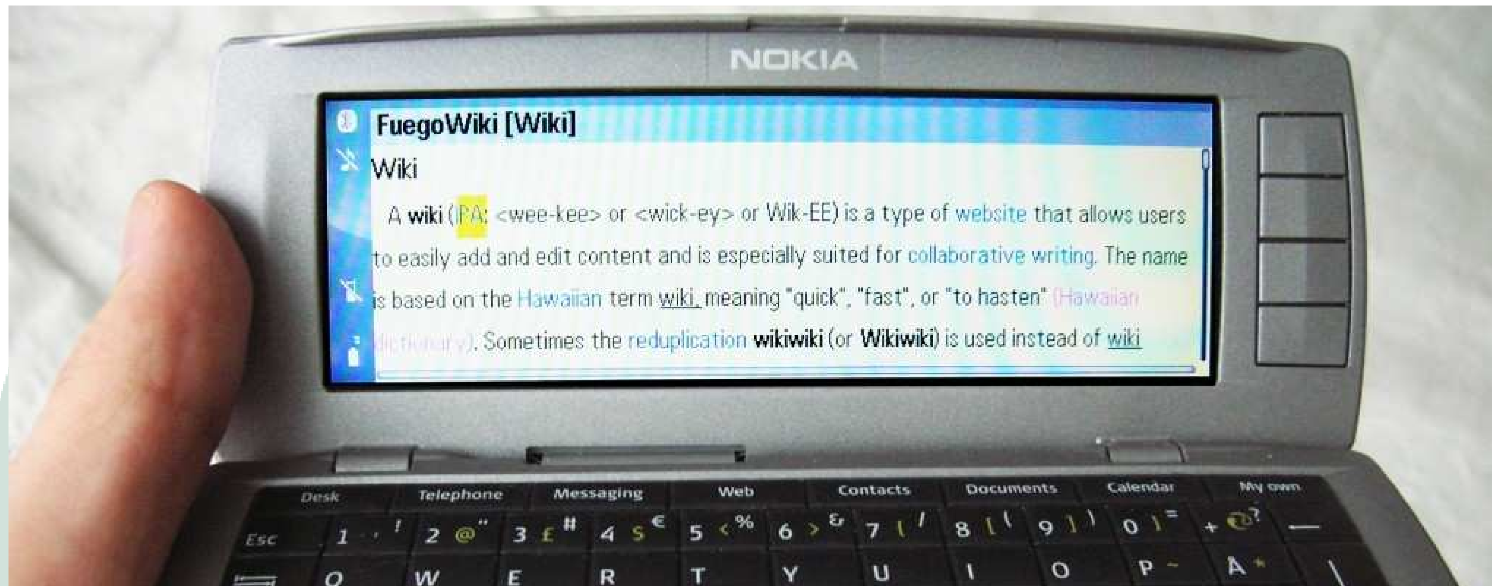
Problem: Too Few Cycles

- Still, **CPU cycles on mobile phones are expensive**
- Even if the phone were fast, cycles eat battery
- Case: Nokia 9500 Communicator
  - Java 300 times slower than my P4 desktop PC
  - Supports >=1Gb RS-MMC storage, but...
  - ...some 10h to parse 1 GB of XML (2min on PC)
- The **Fuego XML Stack** makes your cycles count
- We look at the techniques used in the stack

HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

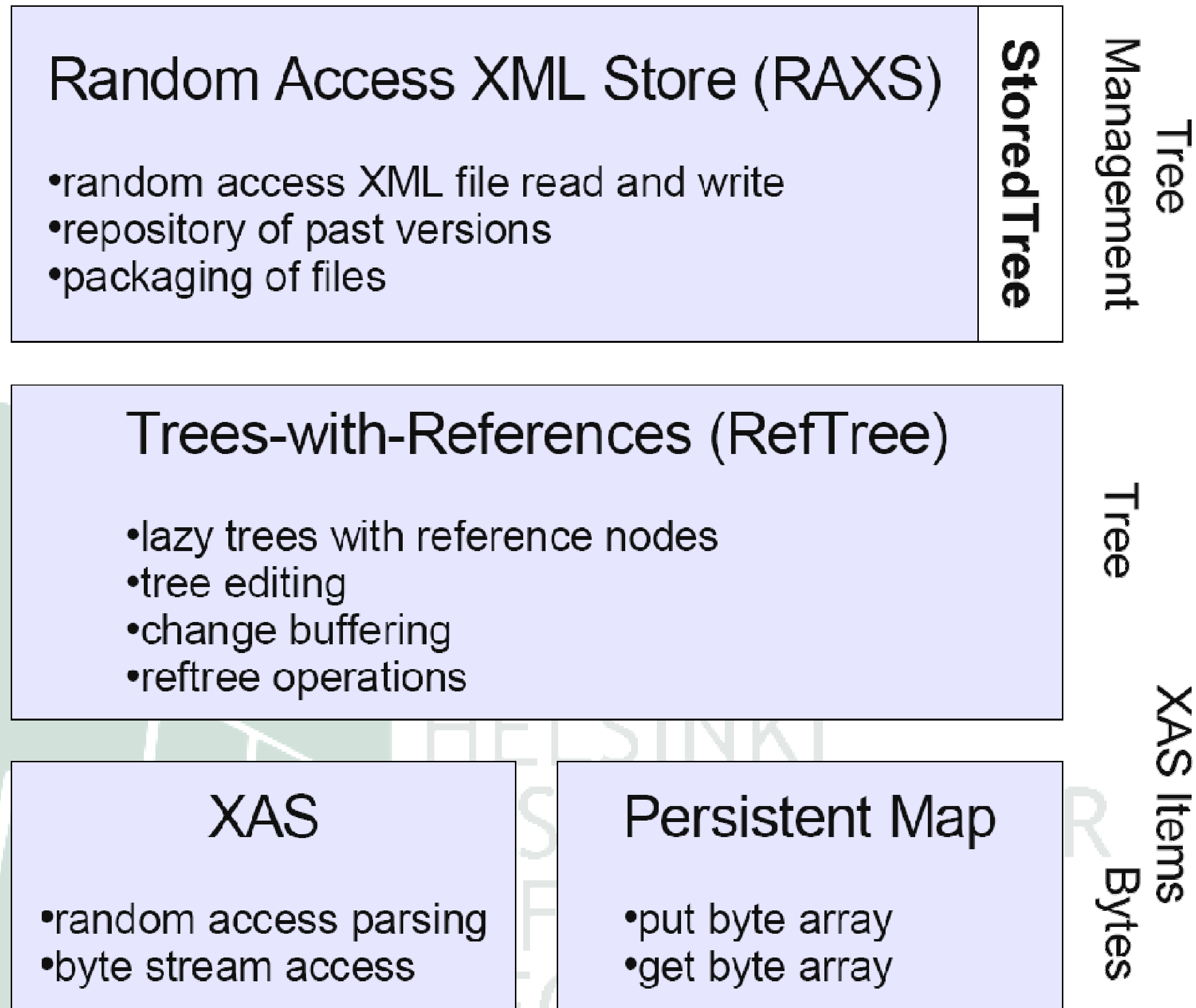
Teaser

- XML editor application running on a Nokia 9500
- Built on the Fuego XML Stack
- XML file being edited (Wikipedia XML dump) is **1GB**



INFORMATION
TECHNOLOGY

The Fuego XML Stack



Fuego XML Techniques

1. Processing XML as a sequence of XML particles
2. Access to XML parser/serializer byte stream
3. Random-access parsing
4. Delayed tree structures
5. Incrementally built mutable tree structure
6. Packaging

   Not presented today:

7. XML Versioning
8. XML Synchronization
9. Alternate serialization format
   - Retain the XML data model, but lose the text format

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

XML as Sequences

- SAX, XmlPull, StAX produce parse "events"
- Similarly, XAS has XML particles known as *Items*

```
<?xml encoding="utf-8" ?>
<root id="1">
  Hello
</root>
```

0: SD()	Start Document
1: SE(root{id=1})	Start Element
2: C(Hello)	Text
3: EE(root)	End Element
4: ED()	End Document

Note: whitespace C() Items not shown

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

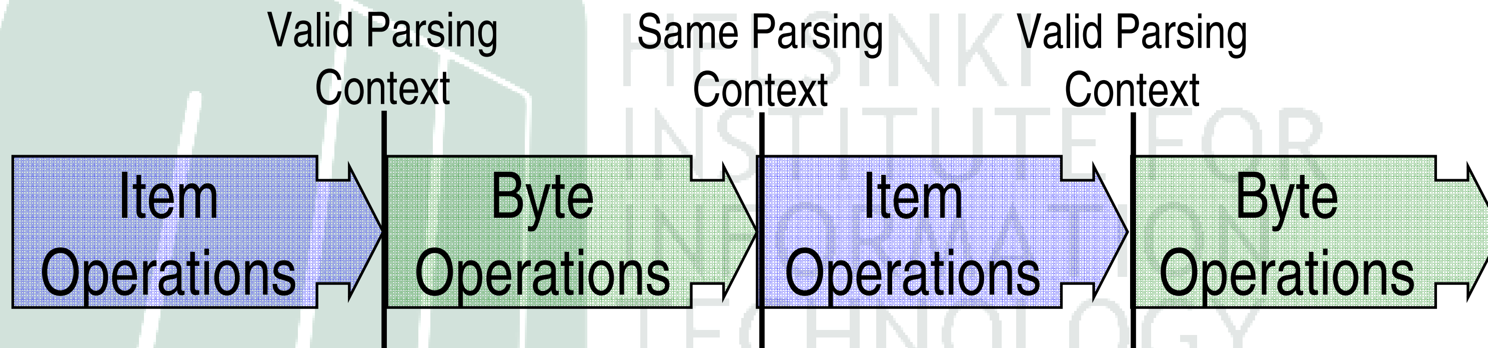
XAS Item Processing

- Process items in a (streaming) linear manner when trees are not needed
  - Less memory (no structure pointers)
  - Simpler code
- Examples
  - XML filtering (remove whitespace, replace tag,...)
  - XML differencing
- XML differencing using XAS Item sequences
  - Align XAS Item sequences using heuristic
  - Alt 1: Output sequence alignment (W3C EXI)
  - Alt 2: Map to matched tree = diff (DocEng 2006)

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

Byte Stream Access

- Some document have huge text nodes
  - E.g. practice of including BLOBS as Base64
- Large subtrees of no interest to application
  - E.g. localized document update
- XAS Byte Stream API provides access to the byte stream beneath the parser/serializer
- *Parsing context* used to ensure valid interaction between layers



Byte Stream Access

- Examples
  - Decode Base64 BLOB
  - Copy document subtree to output
  - Bypass character decode/encode phase
- Currently, we need to know the length in advance
- Most useful when paired with random access parsing and lazy structures (up next...)

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

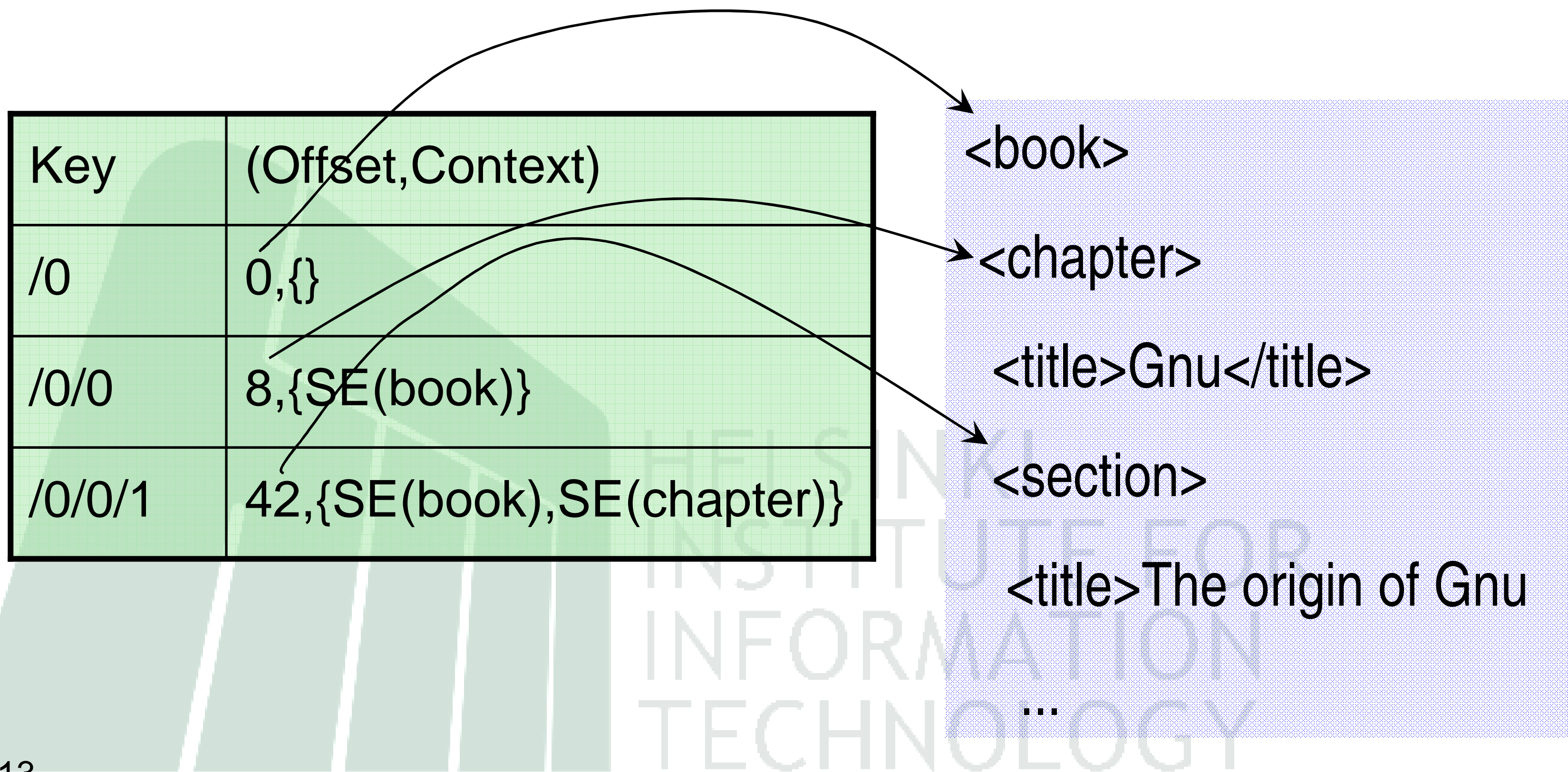
Random Access XML Parsing

- The XAS XML parser can be re-positioned to a new location in its input
- To reposition to a location *p*, we need
  - Offset in input of *p* (and a seekable input)
  - A parsing context for *p*
- Index of user-defined keys and (*offset,parsing context*) is frequently useful

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

Random Access XML Parsing

- Example: DocBook Reader
  - Index `<book>`, `<chapter>`, and `<section>` for instant seek

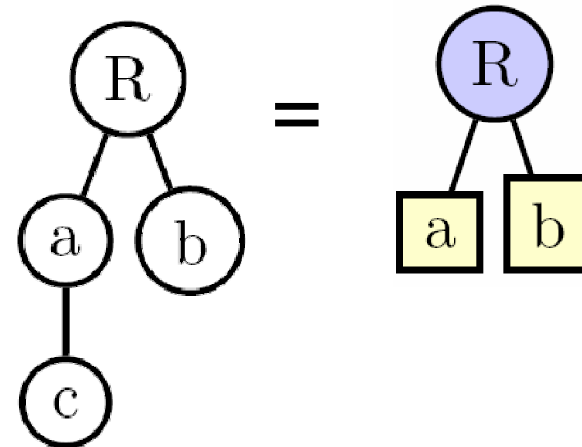


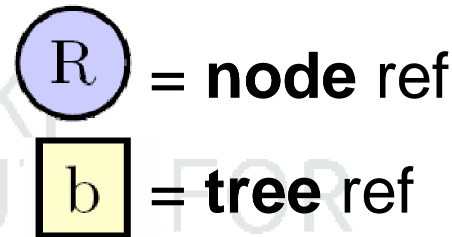
Key	(Offset,Context)
/0	0,{}
/0/0	8,{SE(book)}
/0/0/1	42,{SE(book),SE(chapter)}

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

Lazy Tree Structures (RefTree)

- Use *reference nodes* as placeholders for content from another document
- *Node reference* = placeholder for a single node
- *Tree reference* = placeholder for subtree
- Delayed tree structure = use reference nodes for delayed content
- Explicitly evaluate references using the RefTree API
  - No hidden costs

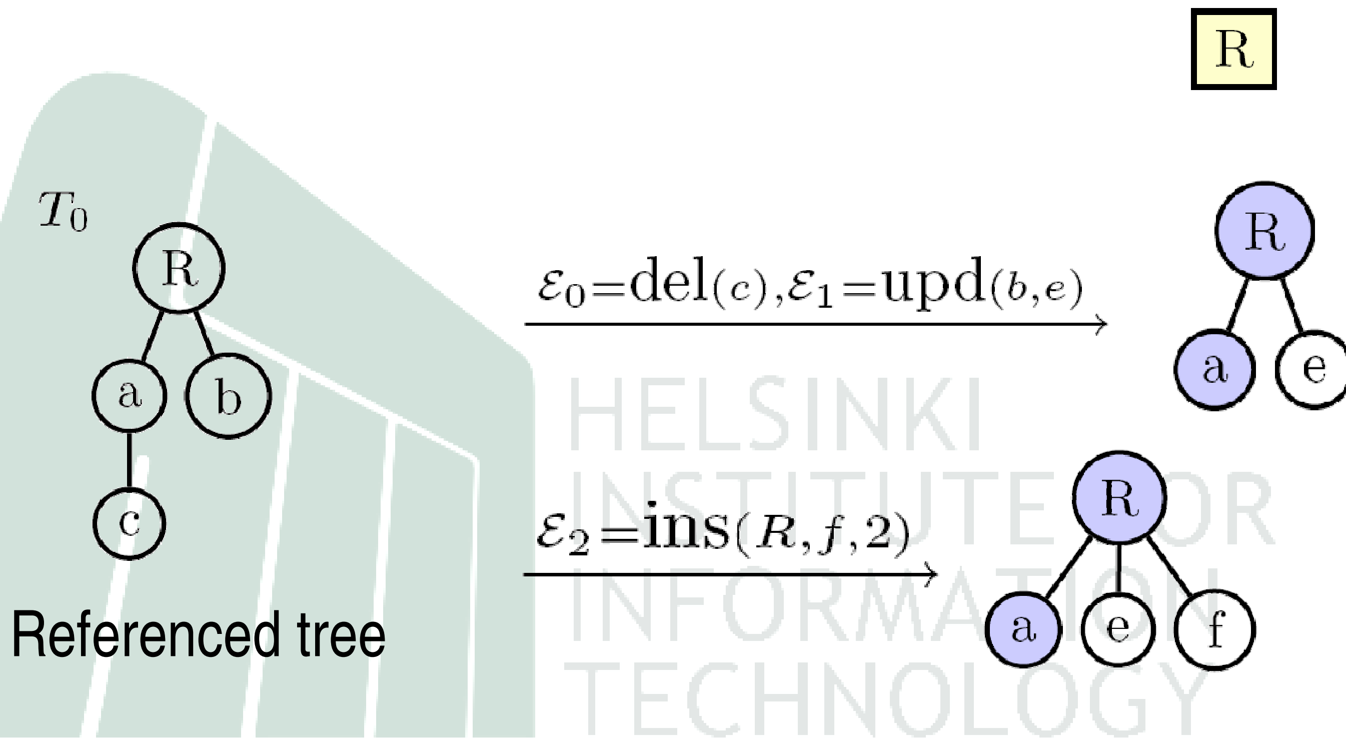




HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

A RefTree as State Change

- A RefTree expresses a set of edits to the tree it references
- When emphasizing this we talk about a *change tree*

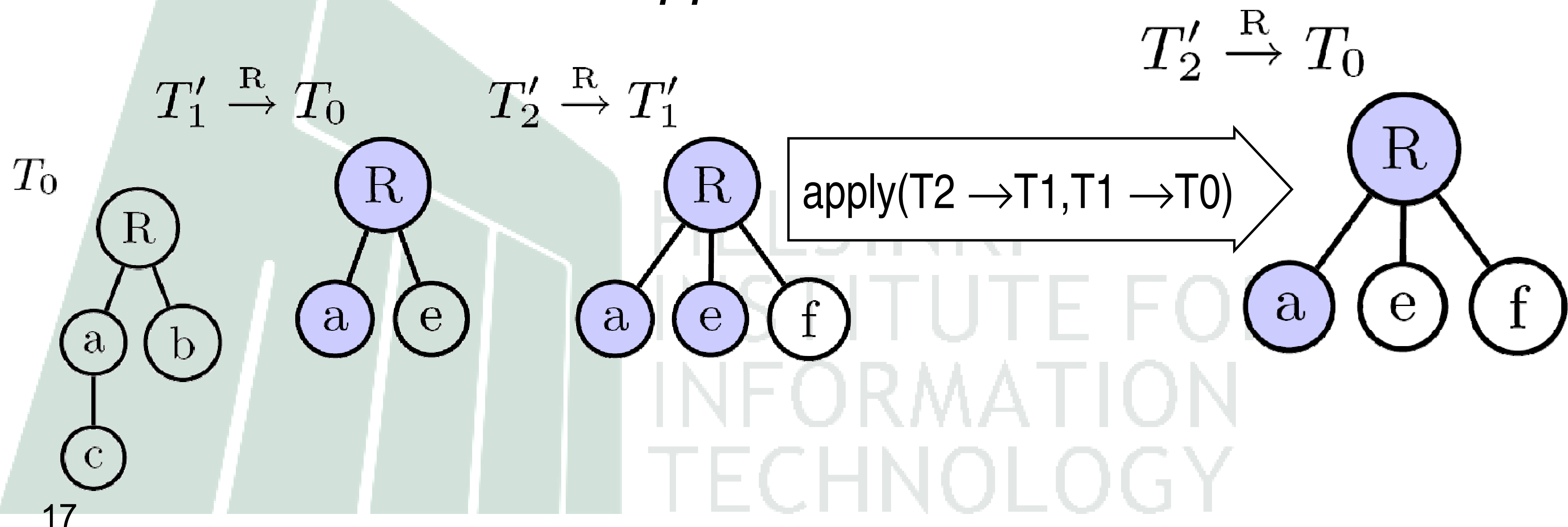


Useful RefTree Operations

- The RefTree API offers some useful primitive operations
- The operations are useful for, e.g., combining edits, reversing edits, and merging
- We look at
  - Application
  - Reference reversal
  - Normalization

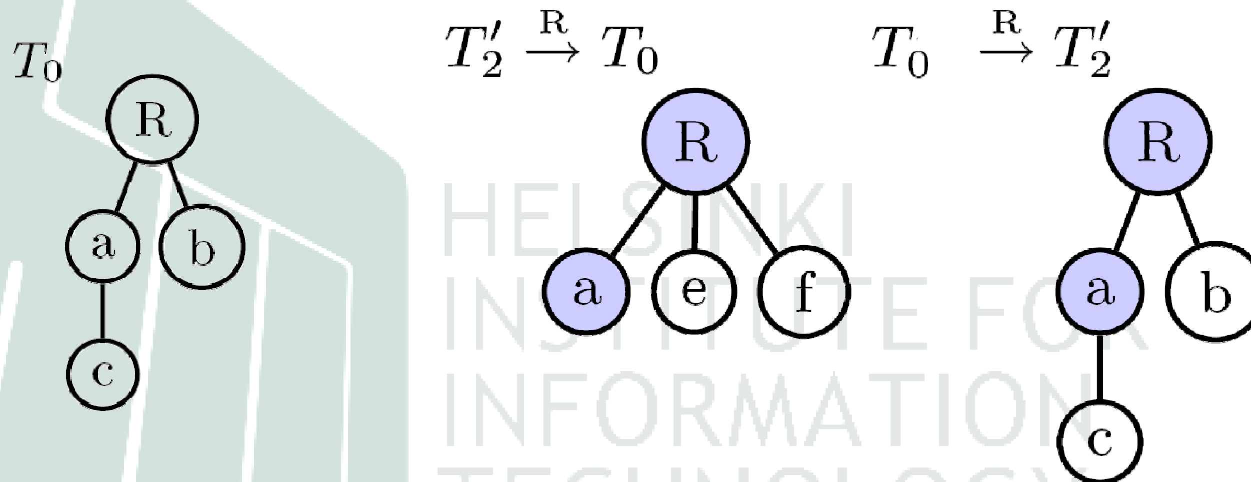
Application of RefTrees

- Notation: $T \rightarrow T_0$ means tree T that references T_0
- We may combine two reftrees $T_1 \rightarrow T_0$ and $T_2 \rightarrow T_1$ to yield $T_2 \rightarrow T_0$
- The tree $T_2 \rightarrow T_0$ is the combined state change of $T_1 \rightarrow T_0$ and $T_2 \rightarrow T_1$
- We call this reftree *application*



Reference Reversal

- We may reverse the roles of trees in $T_1 \to T_2$ by *reference reversal,* yielding $T_2 \to T_1$
- A reference reversal constructs the reverse change tree, i.e. if $T_1 \to T_2$ is the change from state 1 to 2, then $T_2 \to T_1$ is the change from 2 to 1
- Useful in version management



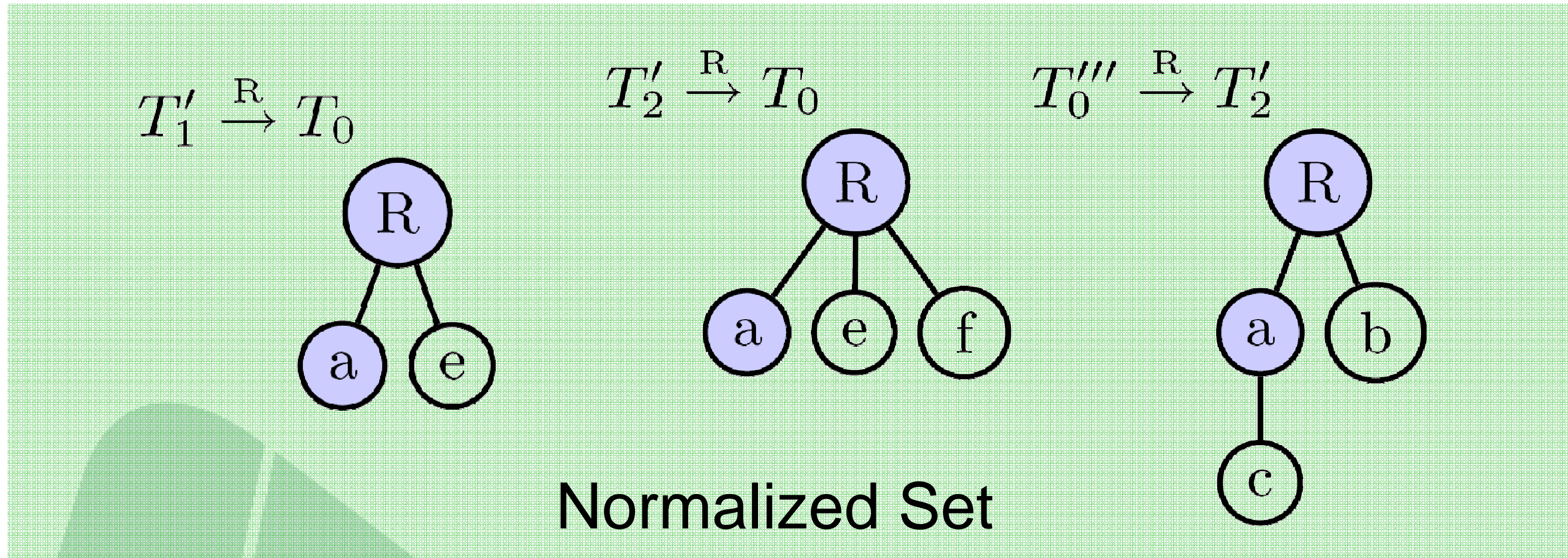
HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

RefTree *Normalization*

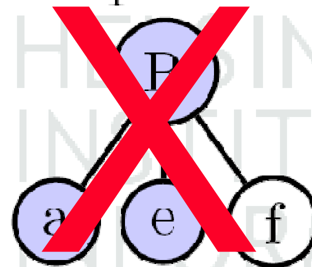
- Start with a set of reftrees referencing a common tree: $\{T_1 \rightarrow T_0, T_2 \rightarrow T_0, T_3 \rightarrow T_0, ...\}$
- In *normalization* we replace tree and node references with equivalent nodes until reference nodes become unique handles to nodes/subtrees in T_0
- In particular, there will be no structural relationship between reference nodes in the trees
- A normalized set of trees can often be processed without knowledge of reference node semantics
- Example: three-way merging

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

RefTree *Normalization*



$T_2' \xrightarrow{R} T_1'$



Because e is a node reference

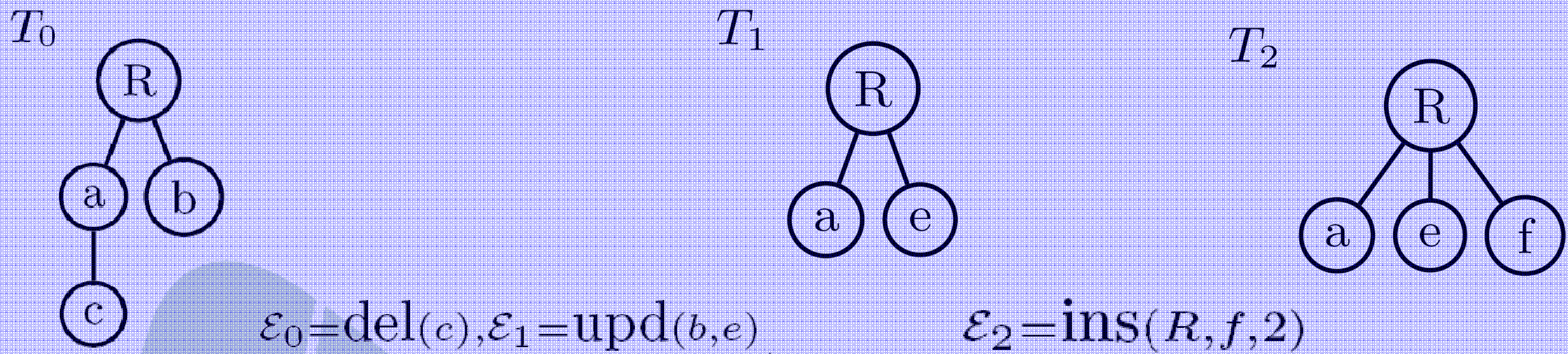
The ChangeBuffer Tree

- *Change buffer* = special mutable tree that sits on top of an immutable base tree
- Initially equal to the base tree
- As edits are made, a change tree expressing the edits is constructed
- The change tree is the only state kept by the change buffer $\rightarrow$
- Huge trees can be edited, as long as the cumulative change tree remains small

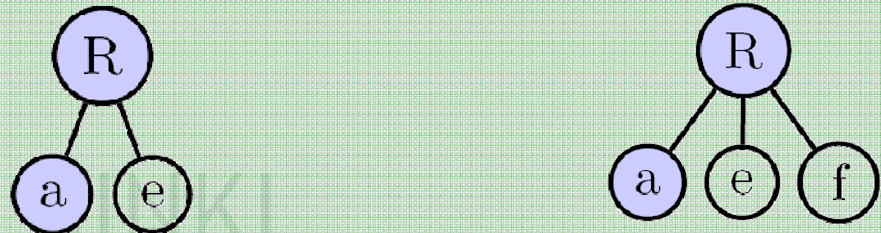
HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

The ChangeBuffer

ChangeBuffer external appearance



R



ChangeBuffer internal change tree

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

Packaging XML with RAXS

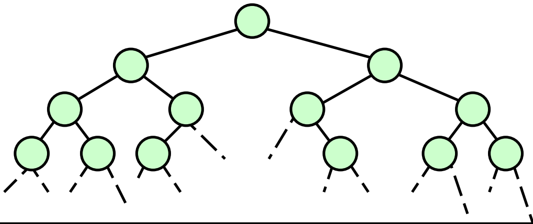
- A common way to handle binary data attached to XML is to use multiple files
  - Seems better than Base64-embedding
- Need to manage XML+satellite files as a single entity
  - for synchronization
  - for easy migration (Open Office uses Zip files)
- RAXS does this in Fuego

HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

Use Case: Editor for Large XML

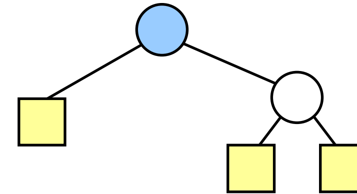
Edits from UI

ChangeBuffer



Maintains

Change Tree



Lazily Constructed RefTree

Loads transient nodes on demand using random access parsing

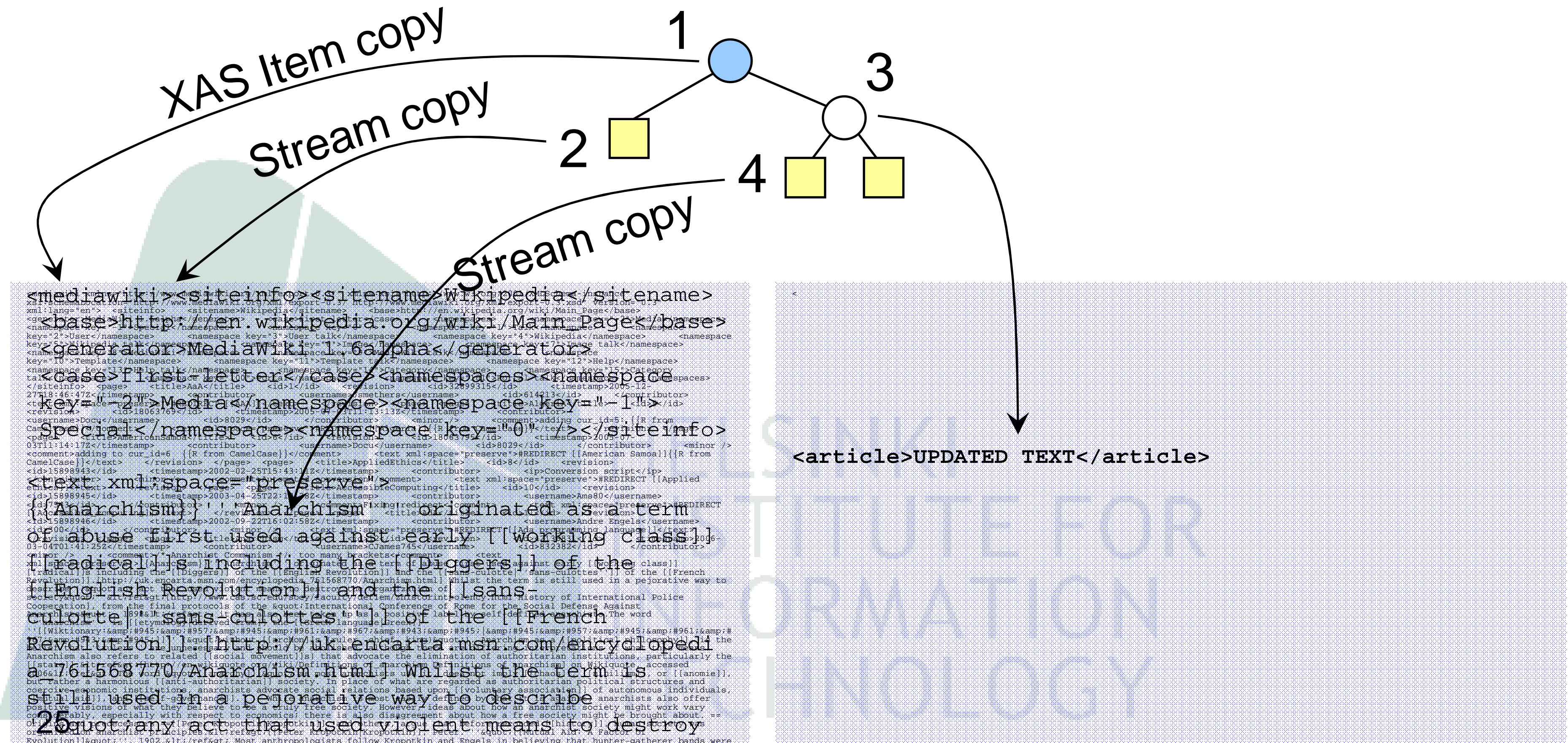
Uses Index

1 GB XML

Key	(Offset,Context)
/0	0,{}
/0/0	8,{SE(book)}
/0/0/1	42,{SE(book),SE(chapter)}
...	...

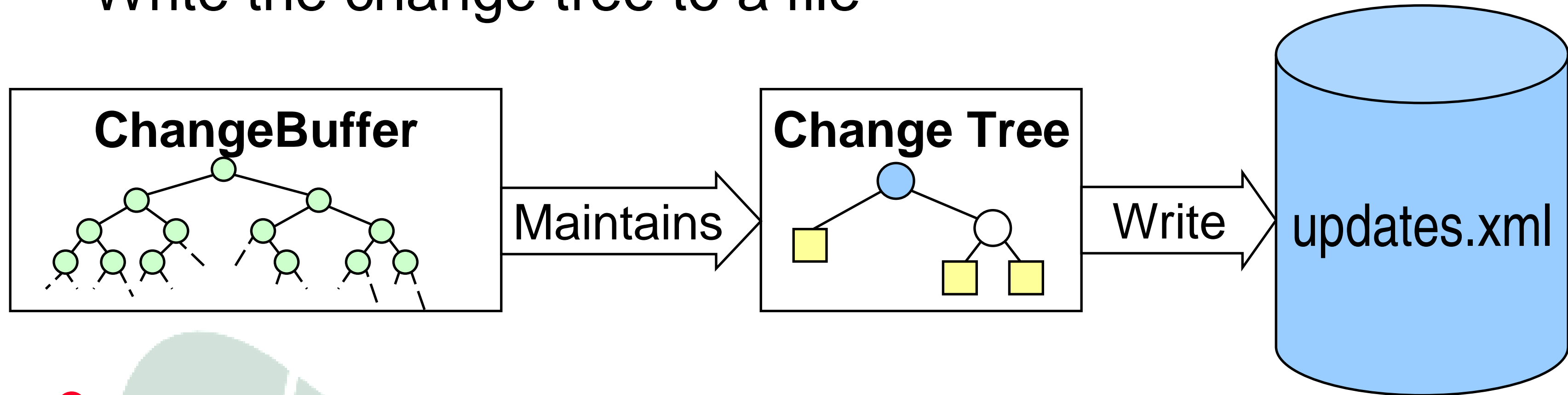
XML File Update (Alternative 1)

- Rewrite the 1 GB file (at close to file copy speed)
- We can use byte stream copy for unchanged subtrees



XML File Update (Alternative 2)

- Write the change tree to a file



- Load changetree into ChangeBuffer on restart

+ Doesn't need any index update (unlike Alt 1)
- Memory usage depends on cumulative edit

HELSINKI
INSTITUTE FOR
INFORMATION
TECHNOLOGY

References

- Comprehensive articles on their way, ask {tancred.lindholm,jkangash}@hiit.fi
- Partial descriptions in
  - ICWS07 (XAS byte API)
  - MobiDE05 (RefTree for dirtrees)
  - DocEng06 (XML diff)
  - PIMRC06 (Middleware)

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY