NPRG039

Advanced Aspects and New Trends in XML (and Related) Technologies

RNDr. Irena Holubová, Ph.D.

holubova@ksi.mff.cuni.cz

Lecture 10. XML Compression

http://www.ksi.mff.cuni.cz/~svoboda/courses/171-NPRG039/

Disadvantages of XML

- Space requirements
  - Often several times higher than in equivalent textual formats
  - Burden for memory/transformation medium
- The overhead associated with processing
  - Tagging, character set, validation, ...
  - The load at the application level

Space Requirements of XML

- Tax for value added by XML
- Decision made by authors of XML
  - „The amount of markup is not important "
  - But: „XML to be directly applicable on the Internet"
- Efforts for simplification of SGML
  - Mechanisms for minimalization of markup
  - Markup freedom
  - Complex implementation

Motivation Example

- Reservation system
  - Components communicate using SOAP

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <t:transaction
       xmlns:t="http://thirdparty.example.org/transaction"
       env:encodingStyle="http://example.com/encoding"
       env:mustUnderstand="true">5</t:transaction>
  </env:Header>
  <env:Body>
    <m:chargeReservation
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
       xmlns:m="http://travelcompany.example.org/">
    ...
```

Motivation Example

```
      ...
      <m:reservation
           xmlns:m="http://travelcompany.example.org/reservation">
        <m:code>FT35ZBQ</m:code>
      </m:reservation>
      <o:creditCard
xmlns:o="http://mycompany.example.com/financial">
        <n:name xmlns:n="http://mycompany.example.com/emp">
            John Smith
        </n:name>
        <o:number>123456789099999</o:number>
        <o:expiration>2007-02</o:expiration>
      </o:creditCard>
    </m:chargeReservation>
  </env:Body>
</env:Envelope>
```

Motivation Example

- Size of message: almost 1 kB
- Processing of multiple messages
  - Communication load
  - System load
- We can use more efficient format:

```
092Z-X2@57G|TID:5||RES|RC:FT35ZBQ|CC|
NM:John Smith|NUM:123456789099999|
EXP:2007-02
```

  - Size: 80B (12x more efficient)
- Consequence:

Motivation Example – Solution

- Compression of XML messages
  - Traditional tools (gzip, bzip2)
  - The reduction is no so efficient
    - Approx. 400B
  - The tools enable to compress various formats
- Both messages contain the same information
  - Theory of information: the messages can be represented using the same number of bits
  - Where is the problem?

Problems with XML Content

- XML documents contain:
  - Metadata
  - Data
  - Padding
- Traditional data compressor can have problems with finding and exploitation of information
- We need special compression techniques
  - XML can be compressed more efficiently than other data formats

Advantages of Special XML Compression

- Better compression ratio
- Direct access to compressed data
  - DOM, SAX
- Querying compressed data
- Less load when processing data
  - Transfer, storing and processing of lower amounts of data
- Faster processing
  - Less I/O operations
  - Data stored in memory

Basics of Data Compression

- Data involve redundant information
  - Repeatable sequences / phrases
  - Non-uniform distribution of symbols
  - …
- Data compression
  - Transformation of data into format which minimizes redundancy
  - Lossy and loss-less compression

Encoding, Compression

- Encoding
  - The process of transformation of data into a particular code representation
  - e.g., Morse alphabet
- Decoding
  - Reconstruction of the original data from the code representation
- Encoder, decoder
  - Algorithms that ensure transformation of data into code representation and its reconstruction
- Compression
  - Encoding for the purpose of reduction of size of data

Code

- Code K is a triple $K = (S, C, f)$
  - S is a finite set of input symbols
  - C is a finite set of code (output) symbols
  - f is a function from S to C^+
- Condition for decodability
  - f must be injective

Prefix Codes

- Important class of codes
- No code word is a prefix of any other code word
- Decodable from left-to-right processing
- Example: Huffman code

  11, 010, 011, 100, 101, 0000, 0001, 0010, 0011
- Non-prefix code:

  100, 101, 110, 111, 1001, …

Entropy

- $S = \{x_1, x_2, \ldots, x_n\}$
- p_i – probability of occurrence of x_i ($1 \leq i \leq n$)
- Entropy of symbol x_i
  - $H_i = -\log_2 p_i$ bits
- An average entropy of a symbol from S
  - $AH(S) = \sum_{i=1..n} p_i H_i = -\sum_{i=1..n} p_i \log_2 p_i$ bits
- Entropy of a message $X = x_{i1} x_{i2} \ldots x_{ik}$ from S^+
  - $H(X) = -\sum_{j=1..k} p_{ij} \log_2 p_{ij}$ bits

Redundancy

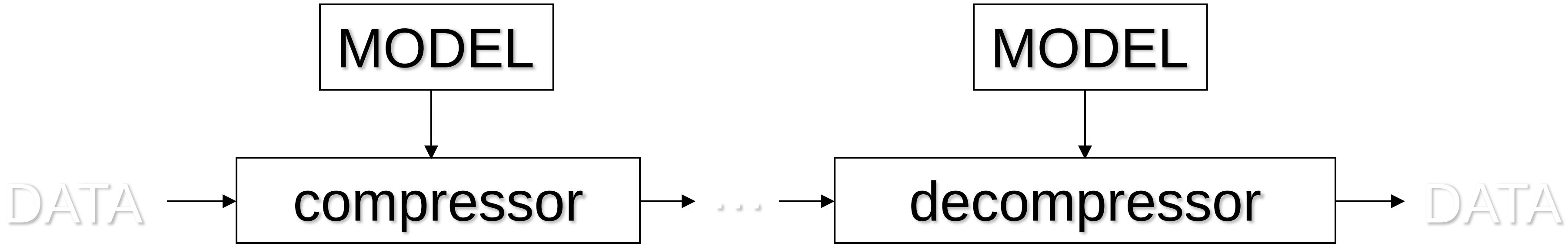
- d_i – number of bits for encoding x_i ($1 \leq i \leq n$)
- The length of code of message $X = x_{i1}x_{i2}\ldots x_{ik}$ from S^+
  - $L(X) = \sum_{j=1..k} d_{ij}$ bits
  - Holds: $L(X) \geq H(X)$
- Average length of code K
  - $AL(K) = \sum_{i=1..n} d_i p_i$ bits
- Redundancy of code K for message X
  - $R(X) = L(X) – H(X) = \sum_{j=1..k} (d_{ij} + p_{ij}\log_2 p_{ij})$ bits
- Average redundancy of code K
  - $AR(K) = AL(K) – AH(S) = \sum_{i=1..n} p_i(d_i + \log_2 p_i)$ bits

Measures of Efficiency of Compressing Algorithms

- Compression ratio
  - Relative value
  - Ratio of the size of input data and encoded data
- Bit rate
  - Absolute value, usually in bits per character (bpc)
  - An average number of bits necessary for encoding of an input symbol
  - Example:
    - size of input symbol: 8 bits
    - compression ratio: 6:1
    - bit rate: 1.33 bpc
- Speed of compression and decompression

Compression Model

- Modelling of the character of input data
  - The more precise, the better compression
- The role of the model during compression
  - Prediction of symbols in the input data
  - Estimation of distribution of probability
- Compressor and decompressor use the same model



Types of Models

- Static model
  - Given beforehand
  - Fixed for all data
  - Example: Morse alphabet
- Semi-adaptive model
  - First walk through – construction of the model
  - Second walk through – data compression
  - The model must be a part of the compressed data
- Adaptive model
  - Construction of the model and compression during a single walk through

Classes of Compression Algorithms

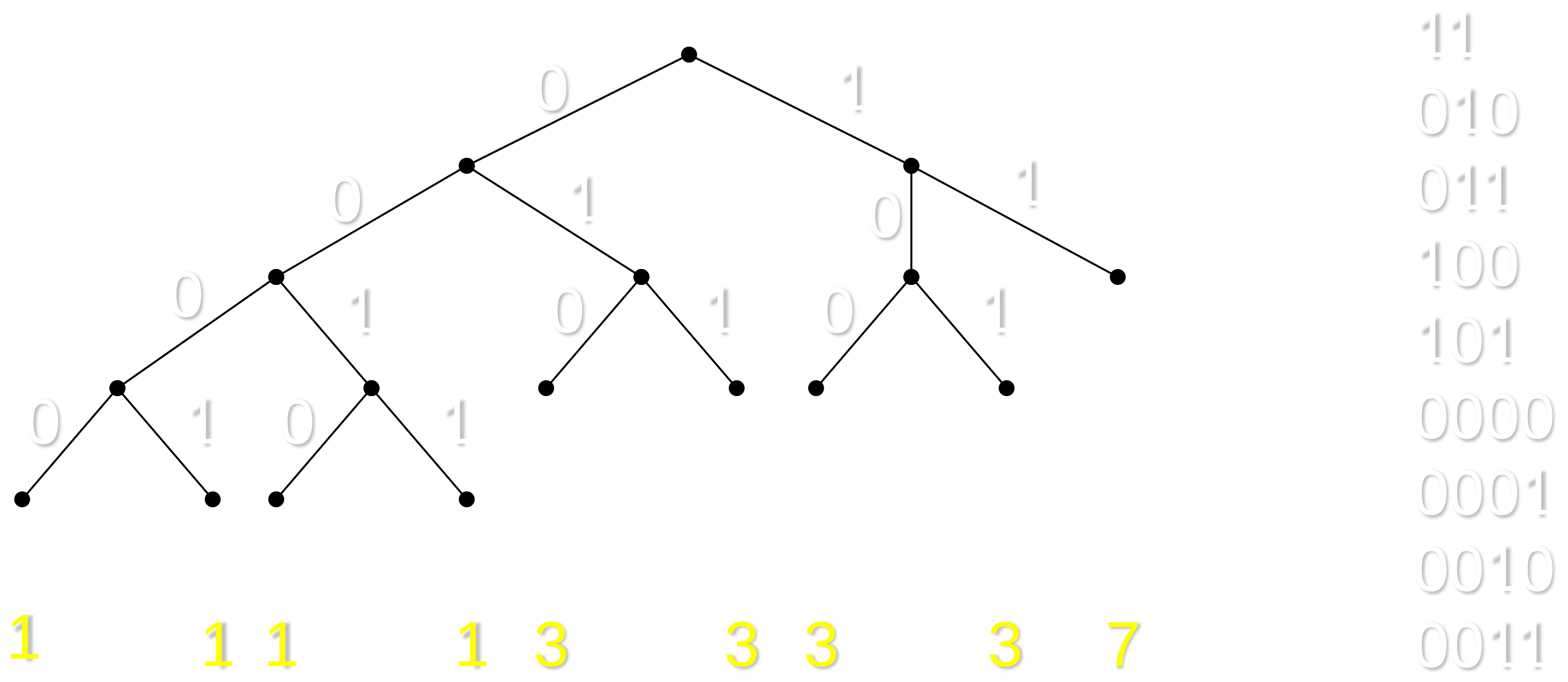
- Statistical compression
  - Probabilistic encoding
  - Huffman encoding
  - Arithmetic encoding
- Dictionary compression
  - Encoding of repeated occurrences of words or phrases
  - Methods with a window (LZ77)
  - Methods with a dictionary (LZ78)

Huffman Encoding

- Words of distinct lengths
  - More frequent symbols = shorter code
  - The code has a prefix property
- Huffman tree
  - Binary tree
  - Edges labeled with 0 and 1
  - Leaves represent symbols to be encoded
  - Code of a symbol = labels on edges from root to leaf
- Static or adaptive construction of a tree

Huffman Encoding – Example

- Alphabet of 9 symbols
- Occurrences of the symbols (1, 1, 1, 1, 3, 3, 3, 3, 7)

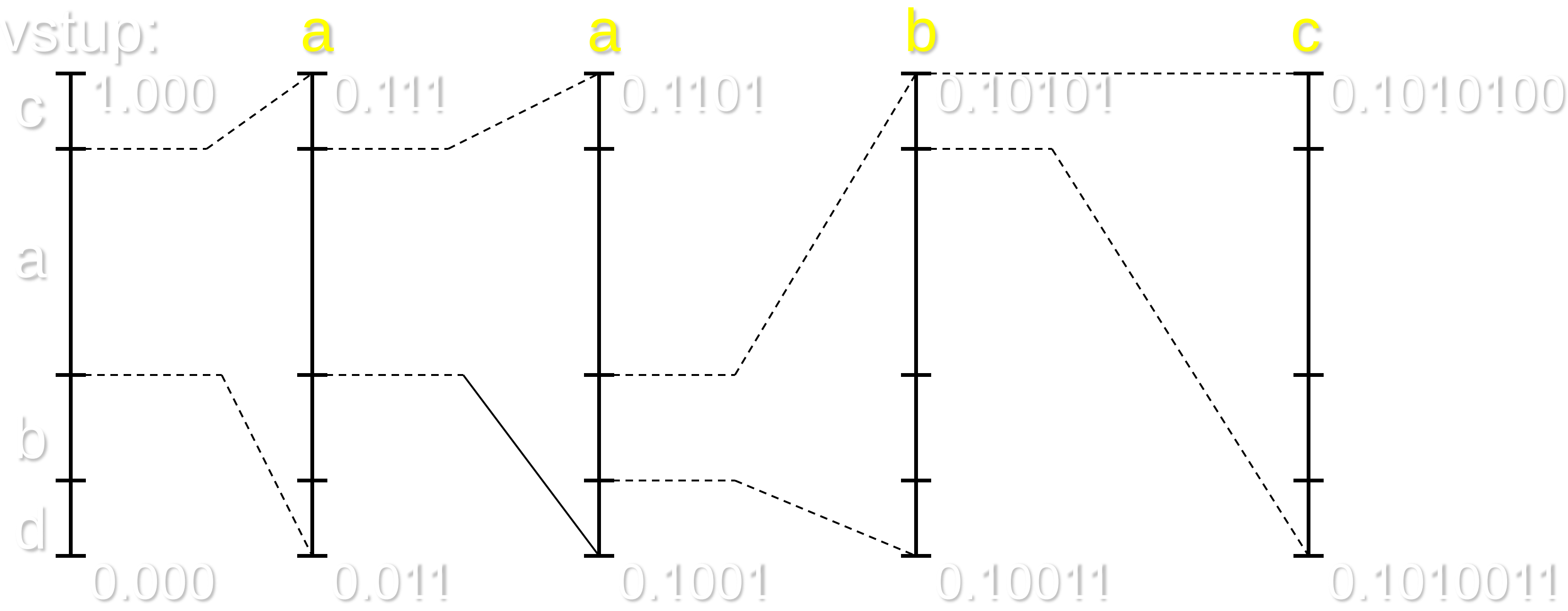


Arithmetic Encoding

- Encoding of a message using an interval of real numbers
  - Initial interval: [0, 1)
- Each symbol refines the initial interval = makes it more specific
  - More probable symbols have lower impact
  - The interval is divided according to cumulative probabilities
- Optimal code
- Problems:
  - Signalization of the end of data
  - Arithmetic of real numbers

Arithmetic Encoding – Example

Symbol	p_i	cp_i	Subinterval
d	0.001	0.000	[0.000, 0.001)
b	0.010	0.001	[0.001, 0.011)
a	0.100	0.011	[0.011, 0.111)
c	0.001	0.111	[0.111, 1.000)



Methods with a Window

- LZ77 a variants
- Sliding window
  - Processed data
  - View
- We search for the longest prefix of the view in the processed data
  - The prefix can overlap the view
  - We encode (distance from the view, length, next symbol)
  - If there is no suitable prefix: (0, 0, next symbol)
  - The window is moved with regard to the length + 1

Methods with a Window – Example

abcbabbabbaab	abacb	abbac…
processed data	view	next data

- We encode string aba: (2, 3, c)
- Window is moved of 3 + 1 = 4 characters

abbabbaababac	babba	ccaab…
processed data	view	next data

Dictionary Methods

- LZ78 and variants
- Dictionary of phrases used in the data
  - The dictionary is extended with new phrases
  - New phrase = prefix (the longest phrase in the dictionary) + next symbol
  - A phrase is encoded as (index of prefix, symbol)
  - If there is no suitable prefix: (0, symbol)
- The dictionary can grow enormously
  - Freezing of dictionary
  - Deleting of parts of dictionary

Dictionary Methods – Example

input	a	b	ab	c	ba	bab	aa	aaa
phrase	1	2	3	4	5	6	7	8
output	(0,a)	(0,b)	(1,b)	(0,c)	(2,a)	(5,b)	(1,a)	(7,a)

Phrase	Index	Code
a	1	a
b	2	b
ab	3	1b
c	4	c
ba	5	2a
bab	6	5a
aa	7	1a
aaa	8	7a

Where We Can Compress XML Data?

- Low-level compression at the I/O level
  - Right before/after I/O operation
  - Suitable for network transfer (compressed HTTP)
- Compression in an external system
  - DBMS
  - Data with XML „view“
- Compression of XML documents
  - Can preserve structure
- Compression in memory
  - Compressed DOM

On-line vs. Off-line Compression

- On-line compression
  - Streaming of XML data
  - Processing of data during decompression
  - SAX interface
- Off-line compression
  - Storing of compressed XML data
  - Usually need to be decompressed before processing
  - Both SAX and DOM interface
  - Querying
  - Space for better compression

Compression and XML Schema

- Compression ignoring schema of XML data
  - Generally usable
  - Worse compression ratio
- Compression based on (exploiting) XML schema
  - DTD, XML Schema
  - Model of XML document
  - Potentially better compression
  - Schema must be available during compression and decompression
- Prediction of structure, differential encoding, …

Note: Canonization of XML document

- First we need a transformation into canonical form (C14N)

```
<?xml version="1.0"?>
<?xml-stylesheet   href="doc.xsl"
   type="text/xsl"   ?>

<!DOCTYPE doc SYSTEM "doc.dtd">

<doc>Hello, world!
<!-- Comment 1 --></doc>

<?pi-without-data     ?>
  <info  author =   "nobody" />
<!-- Comment 2 -->
    <!-- Comment 3 -->
```

```
<?xml-stylesheet href="doc.xsl"
   type="text/xsl"?>
<doc>Hello, world!<!-- Comment 1 --
></doc>
<?pi-without-data?>
<info author="nobody"></info>
<!-- Comment 2 -->
<!-- Comment 3 -->
```

XML Data Compression

- Specialized XML compressors
  - XMill, XMLPPM, XMLZip, XGrind, Millau, …
  - File processing, querying
- XML databases
  - Techniques for minimization of markup
  - Efficient query evaluation
- Binary XML format
  - WBXML
  - Currently no standard
- Proprietary solutions
  - Compressed data formats, …

WBXML

- Compact representation of XML data
  - Part of presentation logic in WAP (Wireless Application Protocol)
- Tokenization of structure on the basis of DTD
  - The data are not compressed
- Coding spaces
  - Space for elements
  - Space for attributes
- Global codes
  - Meaning is independent of actual space
  - 0x00 (end of space), 0x01 (end of element and list of attributes), 0x02 (character entity), 0x03 (string), …
- Simpler and faster processing

WBXML - Example

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="Home" title="The home page">
    <p>Welcome. This is the Home Page.
      <a href="#Product">Click for Product.</a>
    </p>
  </card>
  <card id="Product" title="The Product Card">
    <p>Welcome. This is the page for products.</p>
  </card>
</wml>
```

Byte	Code	Description
1	0x03	WBXML 1.3
2	0x04	Known public ID (WML)
3	0x6A	UTF-8
4	0x00	Length of table of symbols
5	0x7F	wml + content (0x3F + 0x40)
6	0xE7	card + atr (0x27 + 0xC0)
7	0x55	id=
8	0x03	String
9	0x48	H
10	0x6f	o
11	0x6d	m
12	0x65	e
13	0x00	End of string
14	0x36	title=
..	…	…

- WBXML representation
  0x03 0x04 0x6A 0x00 0x7F 0xE7
  0x55 0x03 0x48 0x6F 0x6D 0x65 …

XMill

- Off-line compression
- Separates structure from data
  - Markup and data are processed separately
  - Container compression (gzip)
  - Element and attribute names are compressed using a dictionary
- Grouping of data from the same domain
  - Each element/attribute name has a container
  - Implicit behaviour can be modified
- Semantic compressor
  - Compression of numbers, lists, sequences
- Compression 2x better than gzip with the same speed

XMill – Example

```
<book>
  <title lang="en">Views</title>
  <author>Miller</author>
  <author>Tai</author>
</book>
```

Markup	Code
book	T1
title	T2
@lang	T3
author	T4

- Structural container: T1 T2 T3 C3 / C2 / T4 C4 / T4 C4 / /
- Data containers:

Container	Content
C2	Views
C3	en
C4	Miller, Tai

- On the output the containers are compressed

XMLPPM

- On-line compression
- Combination of several PPM models
  - Prediction by partial matching
  - Multiplex hierarchical modelling (MHM)
  - Element/attribute names, structure of elements, attributes, textual data
- High compression ratio
  - Lower speed

XMLPPM – Example

```
… <title lang="en">Views</title> …
```

- Status of MHM models after processing the XML fragment

	<title	lang=	"en"	>	Views	</title>
Elt	10				FE	FF
Att		0D	en00	FF		
Char					Views00	
Sym		lang00				

XMLZip

- Off-line compression
- The tree is divided at level l
  - The root component contains references at divided subtrees
- The components are compressed using a dictionary
  - Java Zip / Deflate
- Insignificant compression
  - Depends on parameter l
  - The result may be even bigger than the original
- Random access to data
  - DOM with selective decompression of components

XMLZip – Example

```
<bookstore>
  <book>…</book>
  <book>…</book>
</bookstore>
```

- The tree is divided at level l = 2

```
<bookstore>
  <xmlzip id="1"/>
  <xmlzip id="2"/>
</bookstore>
```

```
<book id="1">
…
</book>
```

```
<book id="2">
…
</book>
```

XGrind

- Off-line compression
- Support for simple XPath queries
- Homomorphic compression
  - Preserves the original structure
- Static Huffman code
  - Searching/updating in a compressed domain
  - Checking validity against DTD
  - Necessity for double data walk through
- Markup encoded using a dictionary
  - Construction of the dictionary: Using a DTD or when walking through the data for the first time

XGrind – Example

```
<book>
  <title
lang="en">Views</title>
  <author>Miller</author>
  <author>Tai</author>
</book>
```

Compression output:

```
T0 T1 A0 nahuff(en) nahuff(Views) /
T2 nahuff(Miller) / T2 nahuff(Tai) / /
```

- Before the evaluation the query is encoded
- Decompression is not necessary: queries for equality and the same prefix
- Partial decompression: interval queries and queries for a substring
- Decompression at the level of results

XPress

- Off-line compression
- Support for XPath queries
- Homomorphic compression
- Interval encoding of markup
  - Reverse arithmetic encoding
- Two-pass compression
  - Collecting of statistical information
  - Inference of data types of elements
  - Binary encoding (numbers), static Huffman code (textual data)

XPress – Example

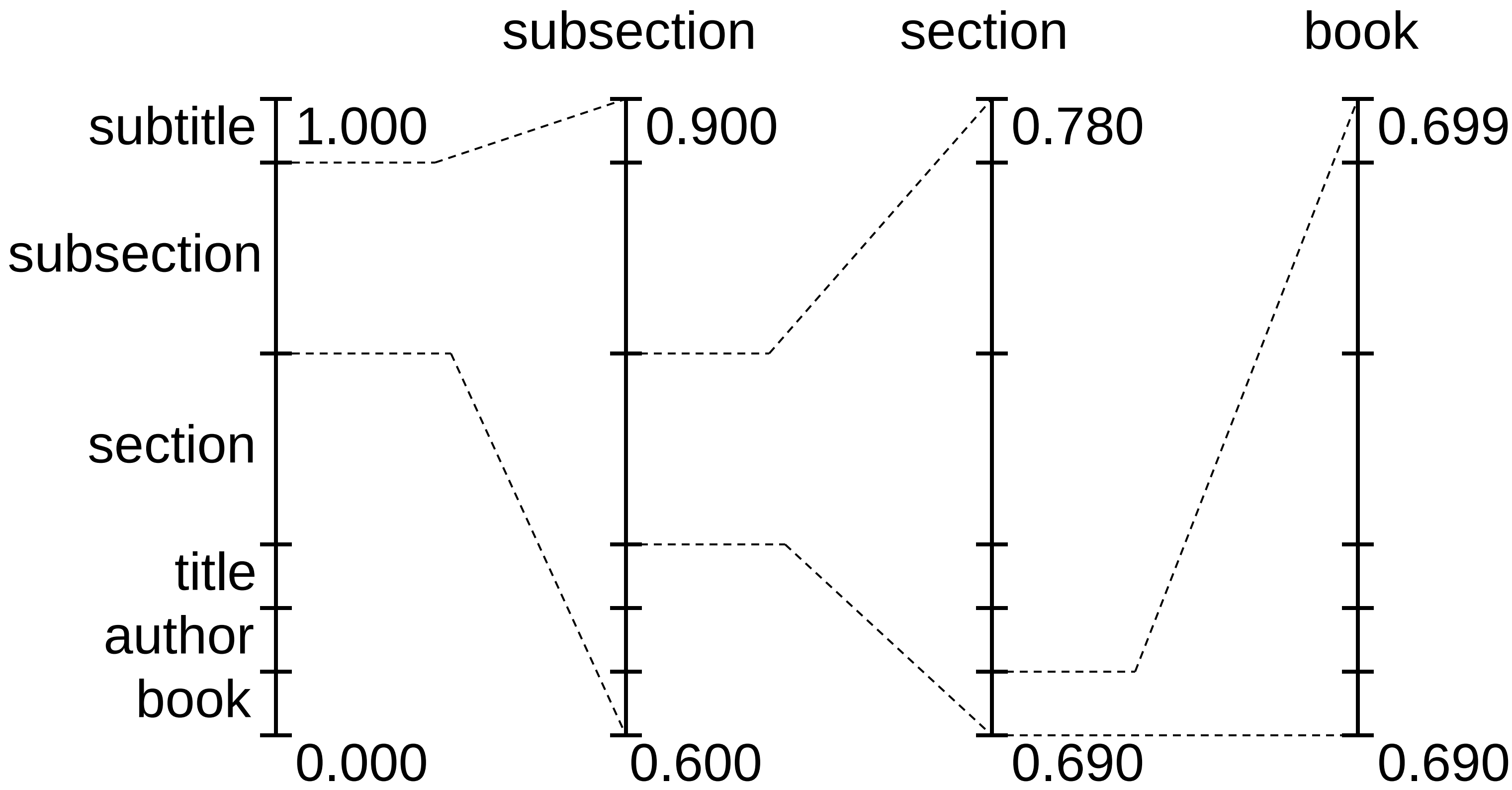
```
<book>
  <author>Author</author>
  <title>Title</title>
  <section>
    <title>Section</title>
    <subsection>

<subtitle>Subsection</subtitle>
      …
    </subsection>
  </section>
</book>
```

Element	p_i	cp_i	Subinterval
book	0.1	0.1	[0.0,0.1)
author	0.1	0.2	[0.1,0.2)
title	0.1	0.3	[0.2,0.3)
section	0.3	0.6	[0.3,0.6)
subsection	0.3	0.9	[0.6,0.9)
subtitle	0.1	1.0	[0.9,1.0)

- Representation of the path to elements using an interval
  - **book.section.subsection**
  - If P_1 is a suffix of P_2, then $I_1 \supseteq I_2$

XPress – Example

- Code of path **`book.section.subsection`**: 0.690



XQueC

- Evaluation of XQuery queries
- Load-driven compression
  - Static Huffman code or ALM
- ALM compressions preserve order
  - $x_1 < x_2 \leftrightarrow comp(x_1) < comp(x_2)$
  - Efficient evaluation of interval queries
- Robust model for storing XML data
  - Supporting data structures
  - Dictionary of markup, tree of the structure of document, summary of the structure, data containers, indexes,…

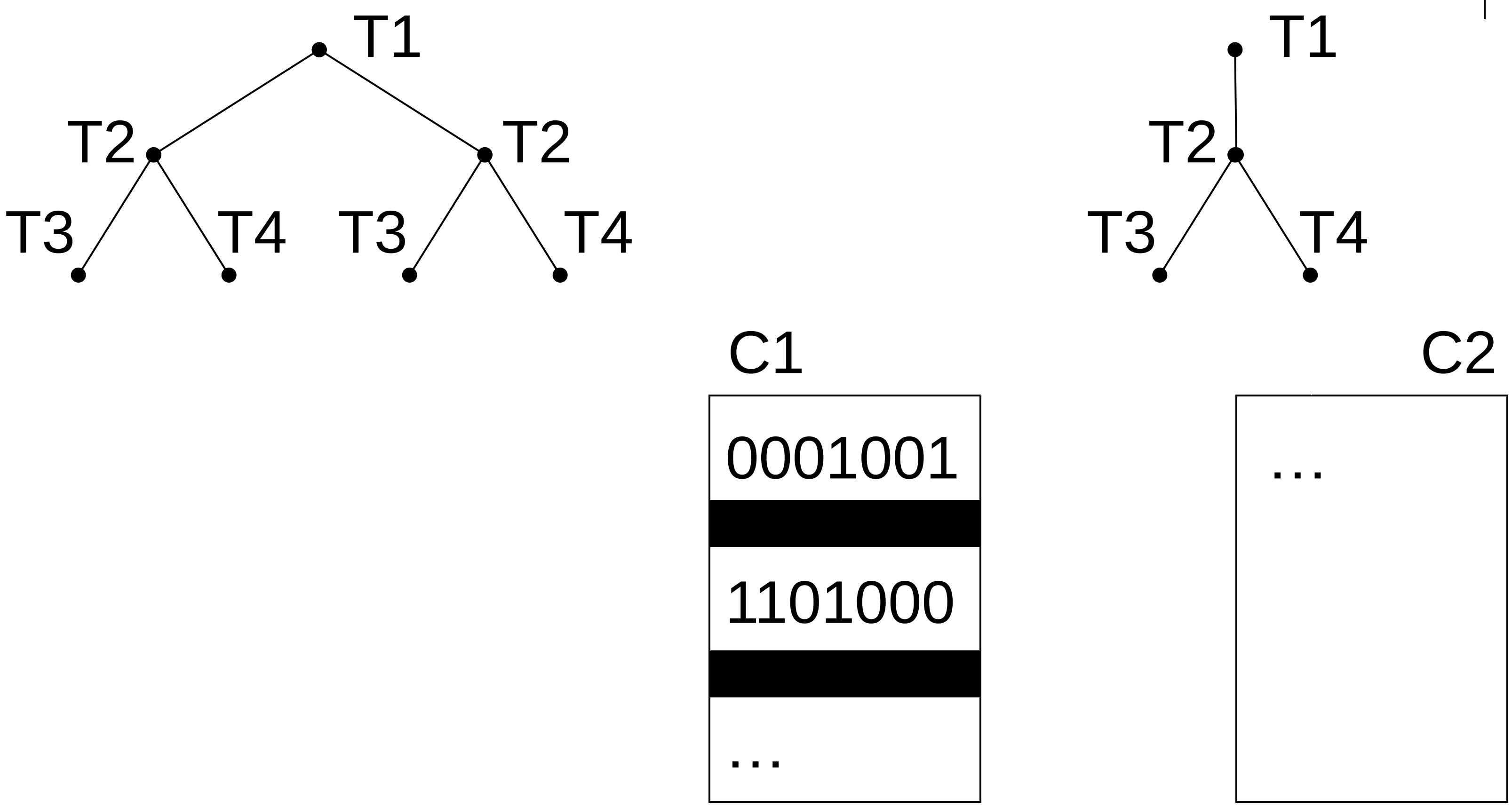
XQueC – ALM Compression

Symbol	Code	Interval
…	…	…
the	c	[thee, therd]
there	d	[there, there]
the	e	[therf, thezz]
ir	b	[ir, ir]
…	…	…
se	v	[se, se]

String	Code
their	cb
there	d
these	ev

- Dictionary method
- The same symbol can have different encoding
- Compression of indexes, …

XQueC – Example



DDOM (Dictionary DOM)

- Compression when processing a document
  - Saving of memory
- Separation of structure from content
- Dictionary compression
  - Markup and data
  - Suitable mainly for data-oriented XML data (small set of values)
  - Data are divided according to elements

DDOM - Example

```
<bib>
  <book>
    <author>Miller</author>
    <author>Tai</author>
    <title>Views</title>
  </book>
  …
</bib>
```

#	Type	Data
…	…	…
2	SE	1
3	SE	2
4	T	0
5	EE	2
6	SE	2
7	T	1
8	EE	2
…	…	..

#	Element
0	bib
1	book
2	author
3	title

#	Text (el:2)
0	Miller
1	Tai

Further Compressors

- Millau
  - Results from WBXML
  - Adds compression of textual datat
  - Streaming of short documents
- XML-Xpress
  - Commercial tool
  - Supports schemas
- Exalt
  - Syntactic compression
  - Adaptive modelling of XML structure
- …

References

- Vojtech Toman: XML Compression – slides, book chapter
- DataCompression.info
  - http://www.datacompression.info
- XMill
  - www.cs.rpi.edu/~sibel/adbs/Papers/sigmod2000-liefke.pdf
- XGrind
  - http://citeseer.ist.psu.edu/tolani02xgrind.html
- XQueC
  - http://www-rocq.inria.fr/~manolesc/PAPERS/XQuecDemo.pdf