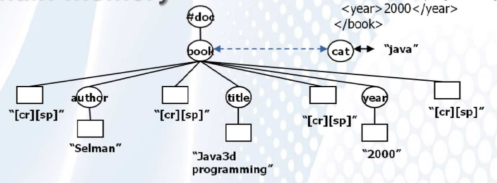
SIXML Version 1.2

Succinct Indexable In-memory Representations for XML Documents

Succinct Indexible XML (SIXML) version 1.2 provides an efficient in-memory representation of large static XML documents, with stable and predictable memory usage, which can be used as a plug-in to create a variety of XML processing APIs. SIXML is based on succinct data structures, which use an information-theoretically minimum amount of space to represent a given data type (see Wikipedia page: http://lra.le.ac.uk/handle/2381/3363).

Representing XML Document In main-memory

```
<book catalogue="java">
  <author>Selman</author>
  <title>Java3d programming</title>
  <year>2000</year>
</book>
```



The benefits of representing an XML document in main memory: Fast navigation and fast data access/modification.

Case Study: Xerces-C 2.8

In Xerces-c the DOM tree is represented using pointers, see Fig 1.1 which shows the 4 pointers required as minimum per node in the DOM tree. Typically node types representations such as DOM_Element and DOM_Text require 416 and 216 bits per node, respectively. The DOM implementation gives a robust API for the DOM tree, at a high memory usage cost, due to the use of pointers.

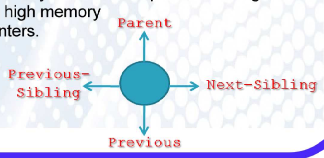


Fig 1.1: DOM Node

Memory Usage

In the table below, we show the space usage of SIXDOM-(CT) compared to Xerces-C, Saxon's TinyTree and a state of the art XML compressor, XMILL. Percentage given is the proportion of the file size.

File	Size	SIXDOM 1.1	Xerces-C	Saxon	SIXDOM-CT	XMILL
Orders.xml	5MB	37%	451%	157%	17%	12%
Lineitem.xml	32MB	28%	399%	161%	13%	5%
XCDNA.xml	607MB	50%	491%	130%	14%	8%

Memory usage typically less than 50% the file size. SIXDOM-CT compresses the text, space usage is even better.

Running times

In the paper [EDBT '08] we provide a comprehensive performance test. Typical results:
Test : Full navigation of document, retrieving all text nodes
Results: SIXDOM 1.1 was ~1.8 times slower than Xerces-C DOM

SIXDOM benefits:

- Very low memory footprint.
- Good for mobile devices.
- Fast processing
- Query-friendly

SIXML 1.2
Fast Parsing,
Plug-in to XML Processing APIs,
Highly space efficient,
Fast navigation and data access.

Pointerless Data Structure

Is there a succinct "(pointerless)" in-memory representation of XML documents, which can give the full XML processing functionality?

We can represent the tree in Figure 1.2 as a parentheses string:

- Document-order numbering.
- Very fast navigation using 2.88 bits/node. [GRRR '06, WEA '06, EDBT '08]

Using parentheses string and other "succinct" building blocks we are able to represent nodes in <14 bits per node, rather than the minimum 216 bits in Xerces-C.

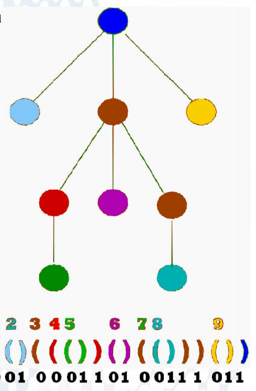


Figure 1.2

SIXDOM 1.1

Provides an highly efficient in-memory representations XML Docs in main memory. Features:
* DOM API (Level 2 and partially 3)
* NodeInfo interface (Saxon)
* C++

We interface SIXML with a DOM API: This we call SIXDOM 1.1

Forthcoming SIXML 1.2

Release Summer 2010

Features:

- Fast, memory efficient parsing. Using the Expat parser (creation of James Clark). Very low memory footprint.
- Cross platform support to languages such as Java and C#.Net.
- Release XML processing APIs based on Succinct data structures.

To maximise the potential of SIXML we are interested in industrial support and partnerships

University of Leicester

O'Neil Delpratt
ond1@mcs.le.ac.uk
(Joint work with Rajeev Raman and Naila Rahman)
Website under construction: http://www.cs.le.ac.uk/SixML/

xmlprague