Succinct Encodings for XPath Location Steps

Jérémy Barbay[1] and S. Srinivasa Rao[2]

[1] David R. Cheriton School of Computer Science
University of Waterloo, Canada.
[2] Computational Logic and Algorithms group
IT University of Copenhagen, Denmark.

Technical Report CS-2006-10
$Date: 2006/04/18 21:52:00 , Revision: 1.22 $

Abstract. We consider in this paper the problem of encoding XML documents in small space while still supporting XPath Location steps efficiently. We model XML documents as multi-labeled trees, and propose for those an encoding which takes space close to the lower bound suggested by information theory, while still supporting the search for the ancestors, descendants and children matching a given label efficiently. To achieve this goal, we extend the previous results from Golynski et al. on strings over large alphabets, and from Barbay et al. on binary relations, to support the rank and select operators in several orders at once; and we extend the results from Geary et al. on ordinal trees to support the DFUDS order, in addition to the other operators.

Keywords: Succinct Encoding, XML, XPath, Location Steps.

1 Introduction

XML [15] is a rapidly emerging format for exchanging data on the web. It standardizes tree structures, so that general tools can be developed and used for the many distinct applications adopting this standard. Among those tools, search engines are prominent, and are strongly based on path navigation. XPath [3, 5] is a language developed for this purpose, which specifies nodes in an XML document by patterns on their rooted path. Each pattern on paths is incrementally described by a sequence of *location steps* , composed of an *axis*, an optional *node test*, and some optional *predicates* . In this paper we model XML documents as multi-labeled trees, and consider the search for ancestors, descendants and children matching a given label, which permits to suppport the location steps described in the definition of XPath.

Several approaches have been studied to solve XPath queries, which can be categorized in two distinct schools of thought. The first one based on a reduction to relational databases: XML documents can be expressed through binary relations, and XPath queries can be answered through queries on those relations (e.g. [9]). This approach benefits from the maturity of the search engines on relational data, but the translation in relational data forbids some trivial and important optimizations [13]. The second school of thought is based on various native implementations of search engines on XML documents (e.g. `Galax` [6] , `Timber` [10, 14]). It draws from both the teachings from relational databases (query optimization path, importance of the formalization) and from the direct access to the tree-structure of the documents (as opposed to the access through relational queries). This development of new tools is the occasion for a paradigm shift in the data structures used: we review two previous contributions, and complement them by several techniques to finally propose an encoding supporting the search among all XPath location axes.

Geary *et al.* [7] proposed an encoding for labeled trees (one label per node) supporting the labeled search for all ancestors, all descendants, and all children in constant time and using $n\big(\lg\sigma + \mathcal{O}(\sigma \lg\lg\lg n/\lg\lg n)\big)$ bits for a document of n tags and σ indexed terms. This space is almost linear in $n\sigma$, and can get too large in practice. Barbay *et al.* [2] proposed a succinct encoding for multi-labeled trees (one or more labels per node) which uses $t\,(\lg\sigma + o(\lg\sigma))$ bits for a tree on n nodes and σ labels, associated in t relations, which supports label-based operators in time $\mathcal{O}(\lg\lg\sigma)$

(a variant of Barbay *et al.* [2]'s encoding uses instead $t(\lg n + o(\lg n))$ bits to support queries in $o(\lg \lg \sigma)$). On single labeled trees, their encoding uses $n\,(\lg \sigma + o(\lg \sigma))$ bits, which is within lower order terms of the lower bound suggested by information theory. Both results claim to have applications to XPath queries on XML documents, but describe only as subset of the operators needed to do so, and index only the tags of an XML document.

Our results are twofold:

- First, we extend the work from Golynski *et al.* [8] on strings and from Barbay *et al.* [2] on binary relations to propose an encoding of strings and binary relations which support the rank and select operators in several different orders (Lemma 1 and Corollary 3).
- Second, we extend the encoding from Geary *et al.* [7] to support the permutation between preorder and DFUDS rank in constant time (Lemma 2), which enables us to propose an encoding for multi-labeled trees which supports the labeled search for all ancestors, descendants, and children (Theorem 1). Our encoding uses $t(\lg \mu\rho + o(\lg \mu\rho))$, which is $t(\lg \rho + o(\lg \rho))$ bits more than the encoding from Barbay *et al.*, where μ is a short notation for $\min(n, \sigma)$, and where ρ is the average number of nodes having the same label on a branch (it has been observed to be very small in practice by Zhang *et al.* [17]).

All our results concerning the running time of operators and algorithms are expressed in the RAM model, where words of size $\Theta(\lg(\max\{n, \sigma\}))$ can be accessed and processed in constant time. All our results concerning the space usage are expressed in the number of bits required, asymptotically in σ, ρ and n.

The paper is organized as follows. In Section 2.2 we show how to support the rank, select and access operators for several orders, while still using essentially optimal space: for succinctly encoded strings in Lemma 1, and for the rows of succinctly encoded binary relations in Corollary 3. In Section 3 we use the results of the previous section to support the search for both ancestors and children of a node matching a particular label. For this we first show how to extend the encoding from Geary *et al.* [7] in order to support the permutation between the preorder and DFUDS ranks in constant time. Given this permutation, the relation associating the nodes of a multi-labeled tree to labels can be succinctly encoded using Corollary 3 in order to support the rank and select operators in both DFUDS and PRE order. We show in Theorem 1 how this can be used to encode multi-labeled trees in order to support the search for all ancestors, children and descendants matching a given label, hence generalizing the results from Barbay *et al.* [2], who supported only the search for the first (in preorder) ancestor and descendant. In Section 4 we outline how our encoding for multi-labeled trees can be used to encode XML documents and support efficiently XPath location steps with simple predicates. While our encoding uses less space than any other representation currently used in practice, it still enables fast searches: we describe how it could be compressed to use even less space, without increasing the time required to search in the document.

2 Multi-order encodings

2.1 Strings

Given a string over an alphabet $[\sigma]$, Golynski *et al.* [8] defined the operators

- $\texttt{string\_rank}(\alpha, x)$, the number of occurrences of α before position x;
- $\texttt{string\_select}(\alpha, r)$, the position of the r-th occurrence of α in the sequence; and
- $\texttt{string\_access}(x)$, the character at position x in the sequence.

We show that it is possible to support, with a negligible space overhead, those operators on any permutation π of the string ($\texttt{string\_rank}_\pi$, $\texttt{string\_select}_\pi$ and $\texttt{string\_access}_\pi$), given a constant time access to $\pi(i)$ and $\pi^{-1}(i)$. We will use this result in Section 3.2 in conjunction with the operators $\texttt{tree\_rank}_{\texttt{PRE}}$, $\texttt{tree\_select}_{\texttt{PRE}}$, $\texttt{tree\_rank}_{\texttt{DFUDS}}(x)$ and $\texttt{tree\_select}_{\texttt{DFUDS}}(r)$ described in Lemma 2.

Lemma 1. *Consider a permutation π on $[n]$, such that the access to both $\pi(i)$ and $\pi^{-1}(i)$ is supported in constant time. A string $S \in [\sigma]^n$ can be encoded using $n(\lg\sigma + o(\lg\sigma))$ bits in order to support the operators* `string_rank` *and* `string_access` *in time $\lg\lg\sigma$ and the operator* `string_select` *in constant time, on* both *S and $\pi(S)$.*

Note that the space required to encode the permutation π is accounted for separately.

Proof (sketch). This is an extension of the results from Golynski *et al.* [8]. Without loss of generality, suppose that $\sigma \leq n$ (the construction is symmetric in the other case). Consider abstractly the string S as a boolean array of size $\sigma \times n$, where column c represents position c of the string and contains a unique one, on the row r corresponding to the character at this position. Supporting the rank and select operators on this matrix is reduced to support them on square matrices of size $\sigma \times \sigma$ via a *domain reduction* [8]. Encode such a matrix M of size $\sigma \times \sigma$ in two strings:

- One string `COLUMNS` on alphabet $[\sigma]$, which lists the columns of each one of M in row major order[3].
- One binary string `ROWDELIM` which gives in unary 0^i the number i of ones in each row, separated by ones.

This encoding completely describes M, and the support of the operators rank and select on S is based only on those two strings.

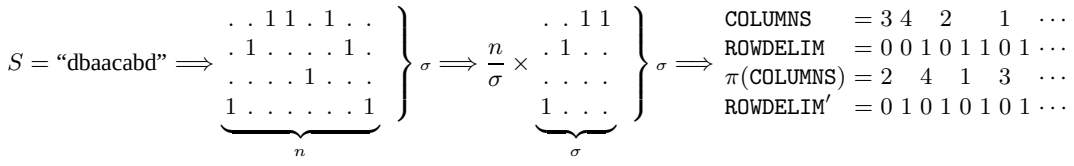

Fig. 1. The relation between the succinct encoding of strings and permutations: the string S is abstracted as a $\sigma \times n$ matrix, which is decomposed into several $\sigma \times \sigma$ matrices. As the matrices have exactly 1 one per column, the list of column numbers corresponding to ones is a permutation of $[\sigma]$. A permutation $\pi = (5, 3, 1, 2, 7, 6, 8, 4)$ of S will be represented by a permutation of `COLUMNS` and a new binary string.

A permutation $\pi(S)$ of S will be represented by the permutation $\pi($`COLUMNS`$)$ of `COLUMNS`, so there is no need to encode it. The string `ROWDELIM`$'$ corresponding to $\pi(S)$ is simply encoded using $n + o(n)$ bit. Then the rank, select and access operators are supported on both S and $\pi(S)$ based only on the three strings `COLUMNS`, `ROWDELIM` and `ROWDELIM`$'$, using in total $n(\lg\sigma + o(\lg\sigma))$ bits. □

2.2 Relations

Given a binary relation $R \in [n][\sigma]$, Barbay *et al.* [2] defi ned the operators

- `label_rank`(α, x), the number of objects labeled α preceding x;
- `label_select`(α, r), the r-th object labeled α, if any, or ∞ otherwise;
- `table_access`(x, α), checks whether object x is associated with label α.

The result from Lemma 1 can easily be extended to those operators on binary relations, provided the adequate defi nition of the permutation of a relation:

Definition 1. *Given a permutation π on $[n]$ and a binary relation $R \in [\sigma] \times [n]$, the* permuted binary relation *$\pi(R)$ is the relation such that $(x, \alpha) \in \pi(R)$ if and only if $(\pi^{-1}(x), \alpha) \in R$.*

[3] The *Row-Major* order lists the elements of the first row, then of the second one, and so on.

Barbay *et al.*'s encoding for binary relations can be extended to support the same operators on both a relation and its permutation. For simplicity, we show here only a restriction of it to the operators on labels (Barbay *et al.*'s encoding support operators on objects too). We prove it in two steps: first for an encoding of size growing with σ, and second for an encoding of size growing with n.

Corollary 1. *Consider a permutation π on $[n]$, such that the access to both $\pi(i)$ and $\pi^{-1}(i)$ is supported in constant time. A binary relation $R \in [\sigma] \times [n]$ of cardinality t can be encoded using $t(\lg \sigma + o(\lg \sigma))$ bits[4] in order to support the operators* `label_rank`, $TableAccess$ *in time $\mathcal{O}(\lg \lg \sigma)$ and the operator* `label_select` *in constant time, on* both *R and $\pi(R)$.*

Proof. We represent the binary relation r through:

- a string $S \in [\sigma]^t$ as the column-major order list of the row of the elements of the relation R,
- two binary strings B and $B_\pi \in \{0,1\}^{t+\sigma}$, listing the cardinalities of each column in unary.

The key issue is to simulate the access to S_π, the column major order list of the row of the elements of the relation $\pi(R)$. To do so, observer that each position p in $\pi(S)$ corresponds to a zero in B_π, and can be decomposed as the number of occurrences p_1 of 1 before it plus the number p_0 of 0 between this position and the last 1 or the beginning of the string, such that $p = p_1 + p_0$ The corresponding position p' in S corresponds to a zero in B defined as the number of occurrences $\pi^{-1}(p_1)$ of 1 before it and the same number p_0 of 0 between this position and the last 1 or the beginning of the string: $p' = \pi^{-1}(p_1) + p_0$.

By Lemma 1, we can support all operators on $\pi(R)$ from S_π and B_π with $o(t)$ additional bits. □

Note that this proof is very simple as it directly transcribes the relation in a string, which is not possible for an encoding which grows in function of n instead of σ, or for an encoding supporting the orthogonal rank and select operators on the columns.

Corollary 2. *Suppose that a permutation π on $[n]$ is encoded such that the access both $\pi(i)$ and $\pi^{-1}(i)$ is supported in time at most $O(\lg \lg n)$. A binary relation $R \in [\sigma] \times [n]$ of cardinality t can be encoded using $t(\lg n + o(\lg n))$ bits in order to support the operators* `label_rank`, $TableAccess$ *in time $\mathcal{O}(\lg \lg n)$ and the operator* `label_select` *in constant time, on* both *R and $\pi(R)$.*

Proof. This time, we need to decompose the relation into σ/n square matrices of size $n \times n$, via the same *reduction step* from Golynski *et al.* [8] and Barbay *et al.* [2], which uses $2\sigma + o(\sigma)$ additional bits and reduces the support of rank and select operators on the lines of the relation R to the rank and select operators on the lines of smaller relations through the definition of rounded rank and select operators.

Consider one of those small relations M, and note t_M its cardinality: as it is of size $n \times n$, Corollary 1 permits to encode it in $t_M(\lg n + o(\lg n))$ bits and still support rank and select on it in time $\mathcal{O}(\lg \lg n)$.

Summing the space used over all smaller relations, the total space used is $t(\lg n + o(\lg n)) + 2\sigma + o(\sigma)$ bits. As we suppose that $\sigma \leq t$, this corresponds to the final result. □

Note that the requirement on the encoding of the permutation π in Corollary 2 is weaker than in Corollary 1. As Corollary 2 supports the operators in time $\mathcal{O}(\lg \lg n)$, it does not matter if computing the permutation's values costs us another $\mathcal{O}(\lg \lg n)$. On the other hand, in Corollary 1 the operators are supported in time $\mathcal{O}(\lg \lg \sigma)$ and an additional cost of $\mathcal{O}(\lg \lg n)$ would be costly.

Corollary 3. *Consider a permutation π on $[n]$, such that the access to both $\pi(i)$ and $\pi^{-1}(i)$ is supported in constant time. A binary relation $R \in [\sigma] \times [n]$ of cardinality t can be encoded using $t(\lg \mu + o(\lg \mu))$ bits in order to support the operators* `label_rank`, $TableAccess$ *in time $\mathcal{O}(\lg \lg \mu)$ and the operator* `label_select` *in constant time, on* both *R and $\pi(R)$, where $\mu = \min(n, \sigma)$.*

[4] Without loss of generality, we suppose that $t > \sigma, n$: it is easy to ignore labels which do not occur in the relation.

3 Trees

3.1 DFUDS order support

The encoding described by Geary *et al.* [7] for ordinal tree is powerful: we extend it to support a different order on the nodes, named DFUDS: it is the order associated to the *depth first unary degree sequence* [4] representation, where all the children of a node are listed before its other descendants (see the Figure 2 for an example).

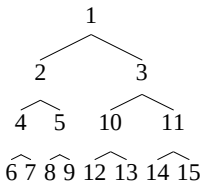

Fig. 2. A binary tree where each node was assigned its rank in the DFUDS order. Note in particular that the DFUDS is *not* the same as the leveled order, here equal to $1, 2, 3, 4, 5, 10, 11, 6, 7, 8, 9, 12, 13, 14, 15$.

Lemma 2. *An ordinal tree on n nodes can be encoded in $2n + o(n)$ bits in order to support in constant time the same operators as Geary* et al. *(in particular the operators* $\texttt{tree\_rank}_{\texttt{PRE}}$, $\texttt{tree\_select}_{\texttt{PRE}}$, $\texttt{tree\_rank}_{\texttt{POST}}$, $\texttt{tree\_select}_{\texttt{POST}}$*), and the operators* $\texttt{tree\_rank}_{\texttt{DFUDS}}$ *and* $\texttt{tree\_select}_{\texttt{DFUDS}}$.

Proof (sketch). The technique is an extension of Geary *et al.*'s to support the depth operator using $o(n)$ bits.

First observe that the DFUDS and PRE ranks of a node x are linked by a simple relation:

$$\texttt{rank}_{\texttt{DFUDS}}(x) = \begin{cases} \texttt{childrank}(x) - 1 + \texttt{rank}_{\texttt{DFUDS}}(\texttt{child}(\texttt{parent}(x), 1)) & \text{if } \texttt{childrank}(x) > 1; \\ \texttt{rank}_{\texttt{PRE}}(x) + \sum_{a \in \texttt{anc}(x) \setminus \{\texttt{r}\}} (\texttt{degree}(\texttt{parent}(a)) - \texttt{childrank}(a)) & \text{otherwise.} \end{cases}$$

As the operators $\texttt{rank}_{\texttt{PRE}}$, parent, childrank and child (see Geary *et al.*'s definition) are all supported in constant time by the encoding from Geary *et al.*, the support of the operator $\texttt{rank}_{\texttt{DFUDS}}$ is reduced to the support of $S(x) = \sum_{a \text{ non root ancestor of } x} (\texttt{degree}(\texttt{parent}(a)) - \texttt{childrank}(a))$ in constant time for each node of the tree.

This support is analogous to the support for depth in constant time [7], using the following property:

Property 1. No node of a mini-tree (resp. micro-tree) except the root has siblings in another mini-tree (resp. micro-tree).

We precompute and memorize $S(x)$ for the root x of each mini-tree, as $n/\max\{\lceil(\lg n)^4\rceil, 2\}$ numbers, each encoded in $\lg n$ bits, which uses in total $o(n)$ bits. We precompute and store the difference $S(y) - S(x)$ between the root x of each micro-tree and the root y of the mini-tree which contains it, as $n/\max\{\lceil(\lg n)/24\rceil, 2\}$ numbers, each encoded in $\lg\lg n$ bits, which uses in total $\mathcal{O}(n\frac{\lg\lg n}{\lg n}) = o(n)$ bits. We finally compute the difference $S(y) - S(x)$ between each node x and the root y of the micro-tree which contains it using a precomputed table using $o(n)$ bits. □

Given the lists of labels of a multilabeled tree in DFUDS order, one can easily support the search for the α-children and α-descendants of a node x, as those are all consecutive in this order. Unfortunately, the DFUDS order is not as practical to support the search for α-ancestors than the preorder. Lemma 2 provides a permutation between the DFUDS order and the preorder, which can be combined with Corollary 3 to support the search for α-ancestors, α-children and α-descendants from a using encoding.

3.2 Multi-Labeled Trees

Using Corollary 3 and Lemma 2, Theorem 1 defines an encoding for multi-labeled trees which uses almost minimal space that supports all the labeled searches at once.

Definition 2 (Barbay *et al.* [2]). *A* **multi-labeled tree** *is an ordinal tree on n nodes together with a set of σ labels, and a set of t pairs from $[n] \times [\sigma]$.*

Barbay *et al.*distinguished in their encoding for each branch the occurrence of each label the closest to the root, by doubling the size of the alphabet. This allow them to find easily the ancestor matching a given label which is the closest to the root, which is sufficient for the task that they are considering.

To efficiently find *all* the α-ancestors of any given node, we encode for each node and for each of its labels α the number of α-ancestors of x. To measure the maximum number of such ancestors, we borrow the notion of recursion levels of XML documents by Zhang *et al.* [17].

Definition 3 (Zhang *et al.* [17]). *Given a rooted path in the XML tree, the maximum number of occurrences of any label minus* 1 *is the* path recursion level *(PRL). The recursion level of a node in the XML tree is defined to be the PRL of the path from the root to this node. The* document recursion level *(DRL) is defined to be the maximum PRL over all rooted paths in the XML tree.*

We extend this notion to a more precise one, which takes into account the relative frequencies of the labels.

Definition 4. *The* recursivity ρ_α of a label α *in a multi-labeled tree is the maximum number of occurrences of α on any rooted path of the tree. The* average recursivity ρ of a multi-labeled tree *is the average recursivity of the labels weighted by the proportion t_α/t of nodes associated with each label α:*

$$\rho = \sum_{\alpha \in [\sigma]} \frac{t_\alpha}{t} \rho_\alpha$$

Note that Zhang *et al.* observed that in practice the DRL is often very small: in the data sets that they considered it is never larger than 10 [17, Table 2]. As the DRL of an XML document is a natural upper bound to its recursivity, we expect the average recursivity of documents to be even lower in practice.

Theorem 1. *Consider a multi-labeled tree on n nodes and σ labels, associated in t relations, of average recursivity ρ. There is an encoding using $t(\lg(\rho\mu) + o(\lg(\rho\mu)))$ bits (where $\mu = \min(n, \sigma)$) that supports*

- *the navigation operators on the structure of the tree in constant time;*
- *the enumeration of the set A of α-ancestors of x in time $\mathcal{O}(\lg\lg\mu + |A| \lg\lg\rho_\alpha)$;*
- *the enumeration of the set D of α-descendants of x α in time $\mathcal{O}(|D| \lg\lg\mu)$;*
- *the enumeration of the set C of α-children of x α in time $\mathcal{O}(|C| \lg\lg\mu)$.*

Proof (sketch). Represent the structure T of the tree as an ordinal tree encoded using the encoding for unlabeled ordinal trees defined by Geary *et al.* [7]: this takes $2n + o(n)$ bits, and supports the navigation operators on the tree structure in constant time as well as the search without node test nor predicate.

Represent the relation R between nodes and labels using Corollary 3 using $t(\lg\mu + o(\lg\mu))$ bits to support the rank, select and access operators on the orders `PRE`, `POST` and `DFUDS` on the nodes in time $\mathcal{O}(\lg\mu)$.

This permits to support the enumeration of all descendants of a node x matching label α in time $\mathcal{O}(|D| \lg\mu)$ using exactly the same technique than Barbay *et al.* [2].

The same technique, used with the DFUDS order, permits to support the enumeration of all children of a node x matching α in time $\mathcal{O}(|C| \lg\mu)$.

As there is no order in which the ancestors of each node are all consecutive, we associate to each label α of a node x the number of ancestors of x matching α: this is a generalization of the trick used by Barbay *et al.* [2] to distinguish

the highest α-node on each rooted branch. For each label α such that $\rho_\alpha > 1$, we represent those numbers in one string $S_\alpha \in [\rho_\alpha]^{t_\alpha}$, where the ith number of S_α corresponds to the ith node labeled α in preorder.

The length of the strings $(S_\alpha)_{\alpha\in[\sigma]}$ is already implicitly encoded in the encoding of R, we need just to encode for each label α its recursivity ρ_α in unary, using $t + \sigma$ bits. To encode the strings themselves, we use the encoding described by Golynski *et al.* [8], which uses $t_\alpha(\lg \rho_\alpha + o(\lg \rho_\alpha))$ bits in order to support the rank and access operator in time $\mathcal{O}(\lg\lg \rho_\alpha)$ and the select operator in constant time.

The total space used by the strings is $\sum_{\alpha\in[\sigma]} t_\alpha(\lg \rho_\alpha + o(\lg \rho_\alpha))$. By concavity of the log, this is smaller than

$$\left(\sum_{\alpha\in[\sigma]} t_\alpha\right)\left(\lg\left(\frac{\sum_{\alpha\in[\sigma]} t_\alpha\rho_\alpha}{\sum_{\alpha\in[\sigma]} t_\alpha}\right) + o\left(\frac{\sum_{\alpha\in[\sigma]} t_\alpha\rho_\alpha}{\sum_{\alpha\in[\sigma]} t_\alpha}\right)\right) = t(\lg\rho + o(\lg\rho)).$$

To support the enumeration of all the α-ancestors of a node x, we first find from R the number p_x of α-nodes preceding x in preorder: this corresponds to a position in S_α, and took us time $\mathcal{O}(\lg\lg\mu)$. Then we initialize i to 1 and iterate as follows: find the position p_i in S_α of the symbol i immediately preceding position p_x: it corresponds to the ith α-node in preorder. If this node is an ancestor of x, output it, increments i and iterates, otherwise stop. Each iteration correspond to a select operation in R (performed in constant time) and some rank and select operators in S_α, hence each is performed in time $\mathcal{O}(\lg\lg\rho)$, hence the total time of $\mathcal{O}(\lg\mu + |A|\lg\rho_\alpha)$ to enumerate the set A of α-ancestors of x.

Summing the space required for T, R and the strings $(S_\alpha)_{\alpha\in[\sigma]}$ gives the final result $t(\lg(\rho\mu) + o(\lg(\rho\mu)))$. □

For algorithms such as the one described by Barbay *et al.* [1, 2], it is useful to search for the first available node matching the location step, where available nodes are defined as those following one particular node in preorder. We show how to extend the encoding of Theorem 1 in order to support those operators:

Corollary 4. *Consider a multi-labeled tree on n nodes and σ labels, associated in t relations, of average recursivity ρ. There is an encoding using $t(\lg(\rho\mu) + o(\lg(\rho\mu)))$ bits (where $\mu = \min(n, \sigma)$) that supports*

- *the navigation operators on the structure of the tree in constant time;*
- *the selection of the first α-ancestor of node x after node y in time $\mathcal{O}(\lg\lg\mu + \lg\lg\rho_\alpha)$;*
- *the selection of the first α-descendant of node x in time $\mathcal{O}(\lg\lg\mu)$;*
- *the selection of the first α-descendant of node x in time $\mathcal{O}(\lg\lg\mu)$;*

Proof (sketch). Without loss of generality, we assume that the operators return ∞ when the answer is not defined (e.g. when the node x has no α-descendant, child or ancestor succeeding to y).

Finding the first descendant and the first child of a node x which matches a label α and succeeds to a node y is already supported by the encoding of Theorem 1. To support the search for the first ancestor after a node y, a naive approach would require to perform a binary search among all the α ancestors of x to find the first one succeeding to y, which would take time $\mathcal{O}(\lg\rho_\alpha\lg\lg\mu)$.

Instead, we add to the encoding of Theorem 1 the support of the operator `LCA` in constant time, either by extending Geary's encoding of the ordinal tree or by spending $2n + o(n)$ bits to encode a second time the structure of the tree as described by Sadakane [12].

Consider a context node x, a threshold node y and a label α. Without loss of generality, we can suppose that y precedes x in preorder (otherwise the operator return ∞), and even that y is an ancestor of x: if it is not then the problem is equivalent to the search for the first α-ancestor of node x after node $\texttt{LCA}(x, y)$.

Using rank and select on the relation R, we can find the first α-descendant z of y in preorder in time $\mathcal{O}(\lg\lg\mu)$. The node z is not necessarily an ancestor of x, but it has the same number i of α-ancestors as y, indicated by the corresponding value in the string S_α, accessed in time $\mathcal{O}(\lg\lg\rho_\alpha)$. Finally, the first α-ancestor of x after y is the α-node corresponding to the value i just before the position corresponding to x in S_α, found in $\mathcal{O}(\lg\lg\mu + \lg\lg\rho_\alpha)$. □

4 Conclusion

We described how to modify Barbay *et al.* [2]'s succinct encoding for multi-labeled trees in order to support the search for child and ancestor matching a given node-test, rather than the descendant and ancestor, and how to extend it in order to support all those operators with the same encoding. This final encoding supports exactly the same operators than the one described by Geary *et al.* [7], but with a new time/space trade-off: while Geary *et al.*'s encoding supports the operators in optimal (constant) time, our encoding uses an amount of space considered asymptotically optimal (i.e. within lower order terms of the amount required to encode the data without supporting any operators on it). We pretend that our encoding is more practical than Geary *et al.*'s: the time in which our encoding supports the operators, $O(\lg \lg \sigma)$, is still manageable; while the space used by Geary *et al.*'s encoding can make it useless in practice as it is growing linearly in $n\sigma$.

This encoding can easily be further extended to support the specificities of XML documents and XPath queries, such as attributes and all location axes (such as children, descendants, attributes,descendants-or-self, preceding-sibling, preceding, or ancestor-or-self), and could be advantageously used in many application (e.g. the work from Josifovski *et al.* [11] and from Zhang *et al.* [16]).

Note also that all the encodings presented in this paper are based on the encoding from Golynski *et al.* [8], which does not compress but supports the select operator in constant time and rank and access in slightly larger time. Most of them make no use of the access operator, and hence can be based instead on the other encoding from Golynski *et al.*, which takes advantage of the entropy of the string to compress the representation, at the cost of the support of the access operator. Of course the entropy of a binary relation or a multi-labeled tree would not be defined in the same way as for a string, but this is an obvious research perspective.

Bibliography

[1] J. Barbay. Adaptive search algorithm for patterns, in succinctly encoded XML. Technical Report CS-2006-11, University of Waterloo, Ontario, Canada, 2006.

[2] J. Barbay, S. S. Rao, J. I. Munro, and A. Golynski. Adaptive searching in succinctly encoded binary relations and tree-structured documents. In *Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching (CPM)*, LNCS, 2006.

[3] A. Berglund, S. Boag, D. Chamberlin, M. F. Fernandez, M. Kay, J. Robie, and J. Simon. XML Path language (XPath) 2.0. Technical report, W3C Working Draft, November 2003. http://www.w3.org/TR/xpath20/.

[4] D. R. Clark and J. I. Munro. Effi cient suffi x trees on secondary storage. In *Proceedings of the Seventh annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 383–391, Philadelphia, PA, USA, 1996.

[5] J. Clark and S. DeRose. XML Path language (XPath). Technical report, W3C Recommendation, November 1999. http://www.w3.org/TR/xpath/.

[6] M. Fernandez, J. Simon, B. Choi, A. Marian, and G. Sur. Implementing xquery 1.0: The galax experience. In *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases*, pages 1077–1080. Morgan Kaufmann, 2003.

[7] R. F. Geary, R. Raman, and V. Raman. Succinct ordinal trees with level-ancestor queries. In *SODA '04: Proceedings of the Fifteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10. Society for Industrial and Applied Mathematics, 2004.

[8] A. Golynski, J. I. Munro, and S. S. Rao. Rank/select operations on large alphabets: a tool for text indexing. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 368–373, 2006.

[9] T. Grust. Accelerating xpath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120. ACM Press, 2002.

[10] H. Jagadish, S. Al-Khalifa, L. Lakshmanan, A. Nierman, S. Paparizos, J. Patel, D. Srivastava, and Y. Wu. Timber: A native XML database. *VLDB*, 11(4):274–291, 2002.

[11] V. Josifovski, M. Fontoura, and A. Barta. Querying xml streams. *The VLDB Journal*, 14(2):197–210, 2005.

[12] K. Sadakane. Succinct representations of lcp information and improvements in the compressed suffi x arrays. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 225–232, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[13] J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D. J. DeWitt, and J. F. Naughton. Relational databases for querying XML documents: Limitations and opportunities. In *The VLDB Journal*, pages 302–314, 1999.

[14] S. P. Shurug. Timber: A native system for querying XML.

[15] F. Yergeau, T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (third edition). Technical report, W3C Recommendation, February 2004.

[16] N. Zhang, V. Kacholia, and M. T. Özsu. A succinct physical storage scheme for effi cient evaluation of path queries in xml. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 54, Washington, DC, USA, 2004. IEEE Computer Society.

[17] N. Zhang, M. T. Özsu, A. Aboulnaga, and I. F. Ilyas. XSEED: Accurate and Fast Cardinality Estimation for XPath Queries. In *to appear Proc. 22nd Int. Conf. on Data Engineering (ICDE) 2006*, 2006.