Compact Ancestry Labeling Schemes for XML Trees*

Pierre Fraigniaud† Amos Korman‡

Abstract

An *ancestry labeling scheme* labels the nodes of any tree in such a way that ancestry queries between any two nodes can be answered just by looking at their corresponding labels. The common measure to evaluate the quality of an ancestry scheme is by its *label size*, that is the maximum number of bits stored in a label, taken over all n-node trees. The design of ancestry labeling schemes finds applications in XML search engines. In these contexts, even small improvements in the label size are important. As a result, following the proposal of a simple interval based ancestry scheme with label size $2 \log n$ bits (Kannan et al., STOC 88), a considerable amount of work was devoted to improve the bound on the label size. The current state of the art upper bound is $\log n + O(\sqrt{\log n})$ bits (Abiteboul et al., SICOMP 06) which is still far from the known $\log n + \Omega(\log \log n)$ lower bound (Alstrup et al., SODA 03). Motivated by the fact that typical XML trees have extremely small depth, this paper parameterizes the quality measure of an ancestry scheme not only by the number of nodes in the given tree but also by its depth. Our main result is the construction of an ancestry scheme that labels n-node trees of depth d with labels of size $\log n + 2 \log d + O(1)$. In addition to our main result, we prove a result that may be of independent interest concerning the existence of a small *universal graph* for the family of trees with bounded depth.

1 Introduction

Background. It is often the case that when people wish to retrieve data from the Internet, they use search engines like Yahoo or Google which provide full-text indexing services (the user gives keywords and the engine returns documents containing these keywords). In contrast to such search engines, the evolving XML Web-standard [2, 37] aims at answering more sophisticated queries. By describing the semantic structure of the document components, it allows users not only to ask full-text queries (e.g., find documents containing the phrase "computer science researches") but also make more sophisticated queries (e.g., find all movies that were released before 1950 by Orson Wells).

To implement sophisticated queries, Web documents obeying the XML standard are viewed as labeled trees, and typical queries over the documents amount to testing relationships between document items, which correspond to *ancestry* queries among the corresponding tree nodes [2, 11, 38, 39]. For instance, the left hand side of Figure 1 depicts the content of an XML document, while the right hand side of the same figure depicts the XML tree associated to that document. These type of trees, together with related external indexes, enables to answer complex queries. For instance, in this toy example, finding all movies that were released before 1950 by Orson Wells can be resolved by checking, for every release dates smaller that 1950, whether it is the one of a movie from Orson Wells. To check the latter, the release date must be a *descendent* of the node "movie", and its sibling node "director" must have the value "Orson Wells".

In fact, to process complex queries, XML query engines often use an index structure, typically a big hash table, whose entries are the tag names in the indexed documents. Due to the enormous size of the Web data and to its distributed nature, it is essential to answer queries using the index labels only, without accessing the actual documents. To allow good performances, a large portion of the index structure resides in the main memory. Since we are dealing here with a huge number of index labels, reducing the length of the labels, even by a constant factor, is critical for the reduction of memory cost and for performance improvement. For more details on XML search engines and their relation to ancestry schemes see, e.g., [1, 3, 8].

Labeling schemes which are currently being used by actual systems are variants of the following simple interval-based ancestry labeling scheme [25, 34]. Enumerate the nodes of an n-node tree according to a DFS traversal starting at the root, providing each node u with a DFS number $DFS(u)$ in the range $[0, n-1]$. Then the label of a node u is the interval $I(u) = [DFS(u), DFS(v)]$, where v is the descendant of u with largest DFS number. An ancestry query then amounts to an interval containment query between the corresponding labels: a node u is an ancestor of a node v if and only if $I(v) \subset I(u)$. Clearly, the size of the

*Supported in part by the ANR project ALADDIN, by the INRIA project GANG, and by COST Action 295 DYNAMO.

†CNRS and Univ. Paris Diderot.

‡CNRS and Univ. Paris Diderot.

```
< art >
    < book >
        < Sutter's Gold >
            < author > Blaise Cendrars < /author >
            < Release date > 1925 < /Release date >
        < /Sutter's Gold >
    < /book >
    < movie >
        < Citizen Kane >
            < director > Orson Wells < /director >
            < Release date > 1941 < /Release date >
        < /Citizen Kane >
        < Once Upon a Time in the West >
            < director > Sergio Leone < /director >
            < Release date > 1968 < /Release date >
        < /Once Upon a Time in the West >
    < /movie >
< /art >
```

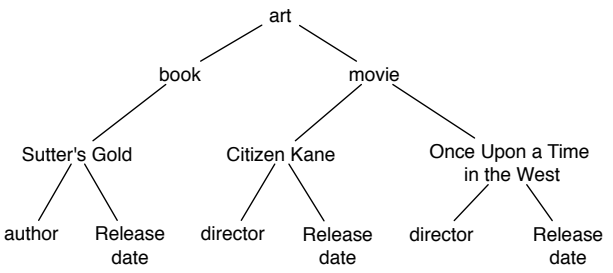

Figure 1: An XML code, and its associated XML tree

produced labels is bounded by $2\log n$ bits[1].

A considerable amount of research has been devoted to improve the upper bound on the label size as much as possible [1, 3, 7, 24, 36]. Specifically, [3] gave a first non-trivial upper bound $\frac{3}{2}\log n + O(\log\log n)$. This was improved the year after to $\log n + O(\sqrt{\log n})$ [7], which is the current asymptotically best upper bound (that scheme is described in detail in the joint journal publication [1]). Independently of that work, an ancestry labeling scheme with larger label size of $\log n + O(\log n/\log\log n)$ was given in [36]. An experimental comparison of different ancestry labeling schemes on XML trees that appear in real life can be found in [24].

The best know upper bound of $\log n + O(\sqrt{\log n})$ is still far from the known $\log n + \Omega(\log\log n)$ lower bound [4], and closing the gab between these bounds remains an intriguing open problem.

On the depth of XML trees. In attempt to provide good performances for real XML instances, we rely on the observation that a typical XML tree has extremely small depth (cf. [8, 10, 30, 29]). For example, by examining about 200,000 XML documents on the Web, Mignet et al. [29] found that *the average depth of an XML tree is 4, and that 99% of the trees have depth at most 8*. Similarly, Denoyer and Gallinari [10] collected 659,338 XML trees taken from the Wikipedia collection, and found that *the average depth of a node is 6.72*.

A natural question is thus to ask what can be achieved if we restrict our attention to trees of small depth. A simple modification of the lower bound proof in [4] gives a lower bound of $\log n + \Omega(\log\log d)$ on the label size required to encode ancestry in trees of depth at most d. However, unfortunately, it is not clear whether one can adapt the techniques from the upper bound proofs in previous schemes to obtain an ancestry scheme that performs well on shallow trees. For example, the simple interval scheme has label size $2\log n$ also for trees with constant depth. As another example, before starting the actual labeling process, the ancestry scheme in [1] first transforms the given tree to a binary tree. This transformation already results with a tree of depth $\Omega(\log n)$, even if the given tree has constant depth. Moreover, previous relevant schemes extensively use and rely on a specific technique, for using *alphabetic codes* on different subpaths. This technique, at least on its surface, does not seem to be more effective on short subpaths, than on long ones.

Our results. Motivated by the fact that the typical XML tree has an extremely small depth, we focus on ancestry schemes whose quality measure is parameterized not only by the number of nodes in the given tree but also by its depth. In our main result we construct an ancestry labeling scheme that labels n-node rooted trees of depth d using labels of size $\log n + 2\log d + O(1)$ bits.

Our labeling scheme uses a novel technique that does not rely on alphabetic codes, as opposed to previous work. Informally, the idea behind our scheme is the following. The labels of the nodes are taken from a small set of integers U, thus ensuring short labels. Each integer in U is associated with some interval taken from some limited range. Similarly to the simple interval based scheme mentioned above, the fundamental rule of our labeling scheme is that a node u is an ancestor of v if and only if the interval associated with the label of u contains the interval associated with the label of v. That way, the ancestry query can be answered very easily, simply by comparing the corresponding intervals. The main technical challenge is to find a way to define and nest these intervals between themselves to be able to appropriately map the nodes of any n-node tree with depth at most d into U, while keeping U small.

Our bound for the label size of ancestry schemes has consequences on the design of implicit representation of graphs (in the sense of [25]). Recall that a graph $\mathcal{U}$ is *universal* for a graph family $\mathcal{G}$ if every graph in $\mathcal{G}$ is an induced subgraph of $\mathcal{U}$. From our ancestry labeling scheme, we can design an adjacency labeling scheme

[1]All logarithms in this paper are taken in base 2.

for n-node trees of depth at most d using labels of size $\log n + 3\log d + O(1)$. Hence, by the equivalence relation stated in [25] between adjacency labeling schemes and implicit representation of graphs, we prove that there exists an $O(nd^3)$-node universal graph for the family of n-node trees with depth at most d.

Other related work. Implicit labeling schemes were first introduced in [25], where an elegant adjacency labeling scheme of size $2\log n$ is established on n-node trees. That paper also notices a relation between adjacency labeling schemes and *universal graphs* (see also [6, 14, 28]). Precisely, it is shown that there exists an adjacency labeling scheme with label size k for a graph family $\mathcal{G}$ if and only if there exists a universal graph for $\mathcal{G}$ with 2^k nodes.

Adjacency labeling schemes on trees were further investigated in an attempt to optimize the label size. In [23] an adjacency labeling scheme using label size of $\log n + O(\sqrt{\log n})$ is presented; and in [6] the label size was further reduced to $\log n + O(\log^* n)$. This current state of the art bound implies the existence of a universal graph of size $2^{O(\log^*(n))}n$ for the family of n-node trees.

Labeling schemes were also proposed for other decision problems on graphs, including distance [4, 16, 17, 19, 26, 27, 28, 31, 35], routing [12, 21, 36], flow [18, 22], vertex connectivity [18, 20], nearest common ancestor [5, 32], tree sibling [4, 13] and various other tree functions, such center, separation level, and Steiner weight of a given subset of vertices [32]. See [15] for a survey on labeling schemes.

2 Preliminaries

Let T be a tree rooted at some node r referred as the *root* of T. The *depth* of a node $u \in V(T)$ is defined as 1 plus the hop distance from u to the root of T. In particular, the depth of the root is 1. The depth of T is the maximum depth of a node in T. For two nodes u and v in T, we say that u is an *ancestor* of v if $u \neq v$ and u is one of the nodes on the shortest path connecting v and r.

A *rooted forest* F is a collection of rooted trees. The depth of F is the maximum depth of a tree in F. For two nodes u and v in F, we say that u is an ancestor of v if and only if u is an ancestor of v in one of the trees in F. Let $\mathcal{F}^{all}$ denote the family of all rooted forests. For integers n and d, let $\mathcal{F}(n)$ denote the family of all rooted forests with at most n nodes, and let $\mathcal{F}(n,d)$ denote the family of all rooted forests with at most n nodes and depth bounded from above by d.

An *ancestry labeling scheme* $(\mathcal{M},\mathcal{D})$ for some family of rooted forests $\mathcal{F}$ is composed of the following components:

1. A *marker* algorithm $\mathcal{M}$ that, given a forest F in $\mathcal{F}$, assigns labels to its nodes.
2. A *decoder* algorithm $\mathcal{D}$ that given two labels ℓ_1 and ℓ_2 in the output domain of $\mathcal{M}$, returns a boolean in $\{0,1\}$.

These components must satisfy that if $L(u)$ and $L(v)$ denote the labels assigned by the marker to two nodes u and v in some rooted forest $F \in \mathcal{F}$, then $\mathcal{D}(L(u),L(v)) = 1 \iff u$ is an ancestor of v in F.

The common complexity measure used to evaluate a labeling scheme $(\mathcal{M},\mathcal{D})$ is the *label size*, that is the maximum number of bits in a label assigned by the marker algorithm $\mathcal{M}$ to any node in any forest in $\mathcal{F}$. Typically, labeling schemes are constructed for the family $\mathcal{F}(n)$, and their label size is bounded by a function of n only. In this paper, however, the label size is bounded by a function of both n and d, the depth of the input trees.

3 A compact ancestry labeling scheme

This section is devoted to proving the existence of an ancestry labeling scheme for $\mathcal{F}(n)$ which assigns labels of size $\log n + 2\log d + O(1)$ to nodes of forests of depth at most d.

Informally, the scheme performs as follows. We construct a set of intervals U such that the nodes of any forest can be mapped to U, in a way that ancestry relation can be answered using a simple interval containment test. I.e., we make sure that for u and v in some forest F, u is an ancestor of v if and only if the interval associated with u contains the interval associated with v. We call such a mapping an *ancestry mapping*. A label of a node in F is simply an interval in U but, in fact, we establish a one-to-one correspondence between the intervals in U and the integers from 1 to $|U|$. Therefore any label can be encoded using $\log|U|$ bits. Thus, to obtain short labels we need U to be small.

The construction of U is done by induction on the number of nodes in the forest. Assume that there exists some set of intervals U_k, such that for any forest F of size at most 2^k there exists an ancestry mapping from F to U_k. Our goal is to find an appropriate set of intervals U_{k+1} that should not be much larger than $|U_k|$, and for which any forest of size at most 2^{k+1} can be embedded to via an ancestry mapping. For simplicity, let us concentrate for now on the case where the given forest is in fact a single tree T of size 2^{k+1}.

A natural approach for a recursive procedure is to choose a separator v whose removal breaks T to disjoint subtrees each of size at most 2^k, and to map each of these subtrees separately to a different copy of U_k. However, this approach would result in a too large

set U_{k+1} (the set U_{k+1} may contain too many vertex-disjoint copies of U_k). Another attempt, would be to group the resulting subtrees to as few as possible vertex-disjoint forests, each of size at most 2^k, and to embed each of them to a different copy of U_k. However, on the one hand, using three copies of U_k would already be too many, and, on the other hand, it may be the case that the subtrees resulting from removing the separator could not be grouped in only two forests each of size at most 2^k. Another difficulty that arises in this recursive approach is that whenever a tree T is split into a collection of subtrees by removing a separator, the subtree containing the root of T plays a special role in the setting of an ancestry scheme, and thus needs to be handled separately.

Our construction is based on a decomposition in which the whole path S from the separator to the root is removed, thus breaking the tree into (1) the path S, and (2) at most d forests. (For each node $u \in S$ we take the forest containing the subtrees hanging down from u's children, excluding the one hanging down from u's child in S, if one exists). After carefully constructing a small set of intervals U_{k+1} (that is not much larger than $|U_k|$), we manage to embed these forests into U_{k+1} via an ancestry mapping. Subsequently, the vertices on the path S from the separator to the root are embedded one by one into U_{k+1} in a way that respects the previous embeddings.

To be able to properly map the (at most) d forests into the small set of intervals U_{k+1} one must pack them very compactly. This is related to the *scale* of the intervals chosen for U_{k+1}. Indeed, each tree and its subtrees use a set of intervals in U_k. This set can be positioned on the real line at different positions, so that to place every tree of the d forests one after the other on the line. To allow flexibility for the choice of the possible positions of each tree, one needs the set U_{k+1} to use refined intervals. However, too refined intervals yields too many such intervals, which in turn results in a too large set U_{k+1}. On the other hand, a loosely refined set of intervals yields too large "gaps" between the trees on the line, which ultimately also results in a too large set U_{k+1}. Determining a good tradeoff between the amount of scaling in the intervals, and the gaps between them, in thus another major issue.

The proof of the theorem below shows how to overcome the above mentioned difficulties. Note that this theorem focusses on the class $\mathcal{F}(n,d)$. In a sense, such a scheme is easier to obtain than a scheme for $\mathcal{F}(n)$ since we can assume that the decoder of the scheme knows d, i.e., given the labels of two nodes u and v in some forest F, the decoder knows that $F \in \mathcal{F}(n,d)$ and therefore that d is a bound of the depth of F. However, by applying a simple trick we show in Corollary 3.1 that such "simpler" schemes can be easily transformed to a scheme for $\mathcal{F}(n)$ with the desired bound.

THEOREM 3.1. *There exists an ancestry labeling scheme for the family of rooted forests in $\mathcal{F}(n,d)$ whose label size is $\log n + 2\log d + O(1)$.*

Proof. For simplicity, we assume n is a power of 2 (if n is not a power of 2, we just round it to the next power of 2, say N, and we add $N-n$ independent nodes to the forest). We begin by defining a set $U = U(n,d)$ of integers, which we use later to label all forests in $\mathcal{F}(n,d)$.

Let $\Gamma_0 = 3n$. We will soon define integers H_i and J_i for each $1 \leq i \leq \log n$. Given these integers, we define for each $1 \leq i \leq \log n$,

$$\Gamma_i = \Gamma_0 + \sum_{j=1}^{i} H_j \cdot J_j.$$

The set of integers U is now defined as the interval

$$U = [1, \Gamma_{\log n}).$$

Informally, the set U can be viewed as composed of $\log n$ layers, where the first layer is $[1,\Gamma_0)$ and for $1 \leq i \leq \log n$, the i'th layer is $[\Gamma_{i-1},\Gamma_i)$. (Note that for $1 \leq i \leq \log n$, the number of integers in the i'th layer is $\Gamma_i - \Gamma_{i-1} = H_i \cdot J_i$.) The set U_k mentioned in the informal description (for embedding forests of size at most 2^k) can be viewed as containing the first k layers, namely $U_k = [1,\Gamma_k)$.

As mentioned, the marker algorithm maps the nodes of any forest $F \in \mathcal{F}(n,d)$ into the integer set U. To perform, the decoder algorithm represents each integer in U as a unique triplet (i,h,j), as follows.

- An integer $\nu \in [1,\Gamma_0)$ is simply represented by $(0,\nu,0)$;
- An integer ν that satisfies $\Gamma_{i-1} \leq \nu < \Gamma_i$ for some $1 \leq i \leq \log n$ can be described as

  $$\nu = \Gamma_{i-1} + hJ_i + j$$

  for unique h and j such that $h \in [0,H_i)$ and $j \in [0,J_i)$; Hence we represent such ν by the triplet (i,h,j).

Observe that given an integer in U, one can easily retrieve its triplet representation and vice versa. Thus, for simplicity of presentation, in the following, we will not distinguish between an integer in U and its triplet representation, unless it may cause a confusion.

Next, we define the promised integers H_i and J_i, $1 \leq i \leq \log n$:

$$H_i = 1 + 3 \cdot n \cdot d \cdot i^2/2^{i-1} \quad \text{and} \quad J_i = \lceil 2 \cdot d \cdot c_i \cdot i^2 \rceil$$

where the c_i's are defined as follows. Let $c_0 = 1$, and, for any i, $1 \leq i \leq \log n$, let

$$c_i = c_{i-1} + 1/i^2 = 1 + \sum_{j=1}^{i} 1/j^2 \ .$$

Note that we have $1 + \sum_{j \geq 1} 1/j^2 < 3$, and hence all the c_i's are bounded from above by 3. Consequently, we obtain $\Gamma_{\log n} = \Gamma_0 + \sum_{j=1}^{\log n} H_j \cdot J_j = O(nd^2)$, and the following claim follows.

Claim 1. $|U| = O(nd^2)$.

Every integer in U is associated with an interval as follows. For $\nu \in [1, \Gamma_0)$, we associate the triplet $(0, \nu, 0) \in U$ with the interval $I_{0,\nu,0} = [\nu]$. Let $x_0 = 1$, and for any i, $1 \leq i \leq \log n$, let

$$x_i = \left\lceil \frac{2^{i-1}}{di^2} \right\rceil .$$

For any $1 \leq i \leq \log n$, any $h \in [0, H_i)$, and any $j \in [0, J_i)$, we associate the triplet $(i, h, j) \in U$ with the interval $I_{i,h,j} = [x_i h,\ x_i(h + j))$. Hence, an interval of level i, $1 \leq i \leq \log n$ (i.e., associated to a triplet (i, h, j) or, equivalently, to an integer ν, $\Gamma_{i-1} \leq \nu < \Gamma_i$, that satisfies $\nu = \Gamma_{i-1} + hJ_i + j$), is of the form $[h, h+j]$ expanded by a factor roughly 2^{i-1}.

We now define a concept of specific interest for the purpose of our proof:

Definition 1. *Let $F \in \mathcal{F}(n, d)$. We say that a mapping $L : F \to U$ is an* ancestry mapping *if, for every two nodes $u, v \in F$ with $L(u) = (i, h, j)$ and $L(v) = (i', h', j')$, we have*

$$u \text{ is an ancestor of } v \text{ in } F \iff I_{i',h',j'} \subseteq I_{i,h,j}.$$

In order to show that there exists an ancestry mapping from every forest in $\mathcal{F}(n, d)$ into U, we use the following notations. For any interval $I \subseteq [1, \Gamma_0)$, let

$$U_0(I) = \{(0, \nu, 0) \mid \nu \in I\}$$

and, for any k, $1 \leq k \leq \log n$, let $U_k(I) = U_0(I) \cup \{(i, h, j) \mid 1 \leq i \leq k, h \in [0, H_i), j \in [0, J_i), \text{and } I_{i,h,j} \subseteq I\}$

Informally, $U_k(I)$ is the set of integers in U that belong to a layer at most k and whose associated interval is contained in I. The following observations are immediate by the definition of the sets $U_k(I)$. Let I and J be two intervals in $[1, \Gamma_0)$. For any k, $1 \leq k \leq \log n$, we have

- $I \cap J = \emptyset \ \Rightarrow\ U_k(I) \cap U_k(J) = \emptyset$,
- $U_k(I) \cup U_k(J) \subseteq U_k(I \cup J)$,
- $I \subseteq J \ \Rightarrow\ U_k(I) \subseteq U_k(J)$,
- $U_{k-1}(I) \subseteq U_k(I)$.

Fix k such that $0 \leq k \leq \log n$. We now give a sufficient condition for the existence of an ancestry labeling scheme using labels in $U_k(I)$. Let I be an interval in $[1, \Gamma_0)$ and let $I_1, I_2, \cdots, I_t$ be a partition of I into t disjoint intervals, i.e., $I = \cup_{i=1}^{t} I_i$ with $I_i \cap I_j = \emptyset$ for any $1 \leq i < j \leq t$. Let F be a forest, and let $F_1, F_2, \cdots, F_t$ be t pairwise disjoint forests such that $\cup_{i=1}^{t} F_i = F$. Using the four properties listed above, one can prove the following.

Claim 2. *If there exists an ancestry mapping from F_i to $U_k(I_i)$ for every i, $1 \leq i \leq t$, then there exists an ancestry mapping from F to $U_k(I)$.*

The following is the main technical ingredient for proving the theorem. Given a graph G, we denote by $|G|$ the number of nodes in G.

Claim 3. *For every k, $0 \leq k \leq \log n$, every forest $F \in \mathcal{F}(2^k, d)$, and every interval $I \subseteq [1, \Gamma_0)$, such that $|I| = \lfloor c_k |F| \rfloor$, there exists an ancestry mapping of F into $U_k(I)$.*

We prove this claim by induction on k. The claim for $k = 0$ holds trivially. Assume now that the claim holds for k with $0 \leq k < \log n$, and let us show that it also holds for $k + 1$.

Let F be a forest of size $|F| \leq 2^{k+1}$, and let $I \subseteq [1, \Gamma_0)$ be an interval, such that $|I| = \lfloor c_{k+1} |F| \rfloor$. Our goal is to show that there exists an ancestry mapping of F into $U_{k+1}(I)$. We consider two cases.

The simpler case is when all the trees in F are of size at most 2^k. For this case, we show a claim stronger than what is stated in Claim 3. Specifically, we show that there exists an ancestry mapping of F into $U_k(I)$ for every interval $I \subseteq [1, \Gamma_0)$ such that $|I| = \lfloor c_k |F| \rfloor$ (instead of $\lfloor c_{k+1} |F| \rfloor$, i.e., a fraction $1/(k+1)^2$ of $|F|$ smaller than what is required to prove the claim). Let $T_1, T_2, \cdots T_\ell$ be the trees in F. We divide the given interval I of size $\lfloor c_k |F| \rfloor$ into $\ell + 1$ disjoint subintervals $I = I_1 \cup I_2 \cdots \cup I_\ell \cup I'$, where $|I_i| = \lfloor c_k |T_i| \rfloor$ for every i, $1 \leq i \leq \ell$. This can be done because $\sum_{i=1}^{\ell} \lfloor c_k |T_i| \rfloor \leq \lfloor c_k |F| \rfloor = |I|$. By the induction hypothesis, we have an ancestry mapping of T_i into $U_k(I_i)$ for every i, $1 \leq i \leq \ell$. The stronger claim thus follows by Claim 2.

Now consider the more involved case in which one of the subtrees in F, denoted by T^*, contains more than 2^k nodes. Our goal now is to show that for every

interval $I^* \subseteq [1, \Gamma_0)$, where $|I^*| = \lfloor c_{k+1}|T^*| \rfloor$, there exists an ancestry mapping of T^* into $U_{k+1}(I^*)$. Once we show this, we can, similarly to the first case, divide the interval I into 3 disjoint subintervals $I = I^* \cup I' \cup I''$, where

$$|I^*| = \lfloor c_{k+1}|T^*| \rfloor \text{ and } |I'| = \lfloor c_k|F'| \rfloor$$

with $F' = F \setminus T^*$. Since we have an ancestry mapping that maps T^* into $U_{k+1}(I^*)$, and one that maps F' into $U_k(I')$, we get the desired ancestry mapping of F into $U_{k+1}(I)$ by Claim 2. (The ancestry mapping of F' into $U_k(I')$ can be done by the induction hypothesis, because $|F'| \leq 2^k$).

For the rest of the proof, our goal is thus to prove the following claim: for every tree T of size $|T|$ with $2^k < |T| \leq 2^{k+1}$, and every interval $I \subseteq [1, \Gamma_0)$, where $|I| = \lfloor c_{k+1}|T| \rfloor$, there exists an ancestry mapping of T into $U_{k+1}(I)$.

Recall that a *separator* of a tree T is a node v whose removal from T (together with all its incident edges) breaks T into subtrees, each of size at most $|T|/2$. It is a well known fact that every tree has a separator. Note however, that there can be more than one separator to a tree. Nevertheless, if this is the case then there are in fact two separators, and one is the parent of the other. In the following, whenever we consider a separator of a rooted tree T, we refer only to the separator of T which is closer to the root.

We make use of the following decomposition of T. We refer to the path S from the separator of T to the root of T as the *spine* of T. (This spine may consist of only one node, namely, the root of T.) Let $v_1, v_2, \cdots, v_{d'}$ be the nodes of the spine S, ordered bottom-up, i.e., v_1 is the separator of T and $v_{d'}$ is the root of T. It follows from this definition that if $1 \leq i < j \leq d'$ then v_j is an ancestor of v_i. A separator is not a leaf if $|T| > 1$, and therefore $1 \leq d' < d$. (Recall that the depth is 1 plus the distance to the root). By removing the nodes in the spine (and the edges connected to them), the tree T breaks into d' forests $F_1, F_2, \cdots, F_{d'}$, such that the following holds for each $1 \leq i \leq d'$:

- in T, the roots of the trees in F_i are connected to v_i;
- each tree in F_i contains at most 2^k nodes.

The given interval I for which we want to embed T into $U_{k+1}(I)$ can be expressed as $I = [a, b)$ for some integers a and b, and we have $b - a = |I| = \lfloor c_{k+1}|F| \rfloor$. For every $i = 1, \ldots, d'$, we now define an interval I_i (later, we will map F_i into $U_k(I_i)$). Let us first define I_1. Let h_1 be the smallest integer such that $a \leq h_1 x_{k+1}$, and let $\bar{h}_1$ be the smallest integer such that $\lfloor c_k|F_1| \rfloor \leq \bar{h}_1 x_{k+1}$. Note that $\bar{h}_1 \geq 1$. We let $I_1 = [h_1 x_{k+1}, (h_1 + \bar{h}_1)x_{k+1})$. Assume now that we have defined the interval $I_i = [h_i x_{k+1}, (h_i + \bar{h}_i)x_{k+1})$ for $1 \leq i < d'$. We define the interval I_{i+1} as follows. Let $h_{i+1} = h_i + \bar{h}_i$ and let $\bar{h}_{i+1}$ be the smallest integers such that $\lfloor c_k|F_{i+1}| \rfloor \leq \bar{h}_{i+1} x_{k+1}$. We let $I_{i+1} = [h_{i+1}x_{k+1}, (h_{i+1} + \bar{h}_{i+1})x_{k+1})$.

Observe that for $1 \leq i \leq d'$, the interval I_i is simply $I_{k+1,h_i,\bar{h}_i}$. Note also that for every $i = 1, \ldots, d'$, we have $\bar{h}_i x_{k+1} < \lfloor c_k|F_i| \rfloor + x_{k+1}$. It follows that the size of I_i is at most $\lfloor c_k|F_i| \rfloor + x_{k+1} - 1$. Thus, since $h_1 x_{k+1} < a + x_{k+1}$, we get that

$$\bigcup_{i=1}^{d'} I_i \subseteq \Big[a,\ a + (d'+1)(x_{k+1} - 1) + \lfloor c_k|T| \rfloor \Big)$$

$$\subseteq \Big[a,\ a + d \cdot (x_{k+1} - 1) + \lfloor c_k|T| \rfloor \Big).$$

Now, since $d \cdot (x_{k+1} - 1) \leq \left\lfloor \frac{2^k}{(k+1)^2} \right\rfloor$, and $2^k < |T|$, it follows that,

$$\bigcup_{i=1}^{d'} I_i \subseteq \left[a, a + \left\lfloor \frac{|T|}{(k+1)^2} + c_k|T| \right\rfloor \right)$$

$$= [a, a + \lfloor c_{k+1}|T| \rfloor) = I.$$

On the other hand, note that for $1 \leq i \leq d'$, I_i contains at least $\lfloor c_k|F_i| \rfloor$ nodes. Therefore, by the fact that, for any i, $1 \leq i \leq d'$, each tree in F_i contains at most 2^k nodes, we get that there exists an ancestry mapping of each F_i into $U_k(I_i)$. We therefore get an ancestry mapping from all F_i's to $U_k(I)$, by Claim 2. It is now left to map the nodes in the spine S into $U_{k+1}(I)$, in a way that will respect the ancestry relation.

For every i, $1 \leq i \leq d'$, let $\widehat{h}_i = \sum_{j=1}^{i} \bar{h}_j$. We map the node v_i of the spine to the triplet $(k+1, h_1, \widehat{h}_i)$.

Let us now show that $(k+1, h_1, \widehat{h}_i)$ is in $U_{k+1}(I)$. First, the fact that $I_{k+1,h_1,\widehat{h}_i} \subseteq I$ follows from the fact that $I_{k+1,h_1,\widehat{h}_i} = \cup_{j=1}^{i} I_j$, and using $\bigcup_{j=1}^{d'} I_j \subseteq I$. It remains to show that $h_1 \in [0, H_{k+1})$ and that $\widehat{h}_i \in [0, J_{k+1})$. Note that,

$$a < 3n \leq \frac{3nd(k+1)^2}{2^k} \left\lceil \frac{2^k}{d(k+1)^2} \right\rceil = (H_{k+1} - 1)x_{k+1}.$$

Therefore, the smallest integer h_1 such that $a \leq h_1 x_{k+1}$ must satisfy $h_1 \in [0, H_{k+1})$. Recall now that for every i, $1 \leq i \leq d'$, $\bar{h}_i$ is the smallest integer such that $\lfloor c_k|F_i| \rfloor \leq \bar{h}_i x_{k+1}$. Thus

$$\bar{h}_i - 1 < \frac{\lfloor c_k|F_i| \rfloor}{x_{k+1}}.$$

It follows that,

$$\sum_{j=1}^{i}(\bar{h}_j - 1) < \sum_{j=1}^{i} \frac{\lfloor c_k|F_j| \rfloor}{x_{k+1}} \leq$$

$$\frac{\lfloor c_k|F| \rfloor}{x_{k+1}} \leq \frac{c_k 2^{k+1}}{x_{k+1}} \leq 2c_k d(k+1)^2.$$

Thus

$$\sum_{j=1}^{i} \bar{h}_j < d + 2c_k d(k+1)^2 \leq 2c_{k+1} d(k+1)^2 \leq J_{k+1}.$$

Therefore $\widehat{h}_i \in [0, J_{k+1})$.

We now show that our mapping is indeed an ancestry mapping. Observe first that, for i and j such that $1 \leq i < j \leq d'$, we have

$$I_{k+1,h_1,\widehat{h}_i} \subset I_{k+1,h_1,\widehat{h}_j}.$$

Thus, the interval associated with v_j contains the one associated with v_i, as desired.

In addition, recall that, for every $i = 1, \ldots, d'$, F_i is mapped into $U_k(I_i)$. Therefore, if $L(v)$ is the triplet of some node $v \in F_i$, then the interval associated with it is contained in I_i. Since $I_i \subset I_{k+1,h_1,\widehat{h}_j}$ for every j such that $1 \leq i < j \leq d'$, we obtain that the interval associated with v is contained in the interval associated with v_j. This completes the proof of Claim 3.

By Claim 3, we get that there exists an ancestry mapping of any $F \in \mathcal{F}(n,d)$ into U. We use this ancestry mapping to label the nodes in F: an ancestry query between two labels can be answered using a simple interval containment test between the corresponding intervals. The stated label size follows, as each node can be encoded using $\log |U|$ bits, and $|U| = \Gamma_{\log n} = O(nd^2)$. This completes the proof of the theorem.

COROLLARY 3.1. *There exists an ancestry labeling scheme for $\mathcal{F}(n)$ such that any node in a forest of depth d is labeled using at most $\log n + 2\log d + O(1)$ bits.*

Proof. We define a labeling scheme $(\mathcal{M}, \mathcal{D})$ for $\mathcal{F}(n)$. Given a forest $F \in \mathcal{F}(n)$ of depth d, let $\hat{d} = 2^{\lceil \log d \rceil}$, i.e., $\hat{d}$ is the smallest integer larger than d which is a power of 2. Obviously, $F \in \mathcal{F}(n, \hat{d})$. Theorem 3.1 tells us that there exists an ancestry labeling scheme $(\mathcal{M}_{\hat{d}}, \mathcal{D}_{\hat{d}})$ for $\mathcal{F}(n, \hat{d})$ which uses labels composed of precisely $L = \log n + 2\log \hat{d} + O(1)$ bits. (By padding enough zeros to the left of the label, we can assume that each label consists of precisely L bits.) The marker $\mathcal{M}$ uses this scheme to label the nodes in F.

The decoder $\mathcal{D}$ operates as follows. Given the labels of two nodes u and v in F, the decoder $\mathcal{D}$ first finds out what is $\hat{d}$ (this can be done, since n and L are known to $\mathcal{D}$, and since $\hat{d}$ is a power of 2), and then uses $\mathcal{D}_{\hat{d}}$ to interpret the relation between u and v. The bound on the size of a label follows as $L = \log n + 2\log d + O(1)$.

In fact, with a slight increase on the label size, one can have an ancestry labeling scheme for the family $\mathcal{F}^{all}$ of all forests (i.e., in such a scheme, given the labels of two nodes in a forest F, the decoder doesn't have bounds on neither the size of F nor on its depth). This is established by the next corollary.

COROLLARY 3.2. *There exists an ancestry labeling scheme for $\mathcal{F}^{all}$ such that any node in a forest of size n and depth d is labeled using at most $\log n + 2\log d + 2\log\log d + O(1)$ bits.*

Proof. Let F be a forest with n nodes and depth d. Let $\hat{n} = 2^{\lceil \log n \rceil}$. We label the nodes of F using the marker of the scheme $(\mathcal{M}_{\hat{n}}, \mathcal{D}_{\hat{n}})$ for $\mathcal{F}(\hat{n})$ given in Corollary 3.1. We show that by adding $2\lceil \log\log d \rceil$ bits to the label of each node in F, one can assume that given a label of a node in F, the decoder knows the value of $\log d$. Clearly, with $\lceil \log\log d \rceil$ bits one can encode the value of $\log d$. However, if the first $\lceil \log\log d \rceil$ bits in each label are devoted for this encoding of $\log d$, then given a label of a node in F, the decoder, that wishes to extract $\log d$, needs to know $\lceil \log\log d \rceil$ (to distinguish the bits used for encoding $\log d$ from the rest of the bits in the label). One way to tackle this is simply to write zeros in the first $\lceil \log\log d \rceil - 1$ bits and 1 in the following bit, and then to use the following $\lceil \log\log d \rceil$ bits to encode $\log d$. (This method uses $2\lceil \log\log d \rceil$ bits for letting the decoder know $\log d$, however, one can see that using a recursive encoding one can accomplish that using even fewer bits.)

Now given a label of a node in F, the decoder can extract $\log \hat{n}$ (using the size of the label, in a method similar to one described in the proof of Corollary 3.1). Since $\hat{n} = 2^{\log \hat{n}}$, we can assume that the decoder knows $\hat{n}$. Thus, to extract the ancestry relation between the two nodes in F, the decoder uses $\mathcal{D}_{\hat{n}}$.

4 Adjacency labeling scheme and implicit representation of trees

The ancestry labeling scheme described in the previous section can be advantageously transformed into an adjacency labeling scheme which is very efficient for trees of small depth. Recall that an *adjacency labeling scheme* for the family of graphs $\mathcal{G}$ is a pair $(\mathcal{M}, \mathcal{D})$ of marker and decoder, satisfying that if $L(u)$ and $L(v)$ are the labels given by the marker $\mathcal{M}$ to two nodes u and v in some graph $G \in \mathcal{G}$, then: $\mathcal{D}(L(u), L(v)) = 1 \iff$ u and v are adjacent in G..

Similarly to the ancestry case, we evaluate an adjacency labeling scheme $(\mathcal{M}, \mathcal{D})$ by its *label size*, namely the maximum number of bits in a label assigned by the marker algorithm $\mathcal{M}$ to any node in any graph in $\mathcal{G}$.

For two nodes u and v in a rooted forest F, u is a parent of v if and only if u is an ancestor of v and $depth(u) = depth(v) - 1$. Also, u is a neighbor of v if and only if either u is a parent of v or v is a parent of u. It follows that one can easily transform any ancestry labeling scheme for $\mathcal{F}(n, d)$ to an adjacency labeling scheme for $\mathcal{F}(n, d)$ with an extra additive term of $\lceil \log d \rceil$ bits to the label size (these bits are simply used to encode the depth of a vertex). The following theorem follows.

THEOREM 4.1. *There exists an adjacency labeling scheme for $\mathcal{F}(n)$ such that any node in a forest of depth d is labeled using $\log n + 3 \log d + O(1)$ bits.*

COROLLARY 4.1. *There exists an adjacency labeling scheme for $\mathcal{F}^{all}$ such that any node in a forest of size n and depth d is labeled using at most $\log n + 3 \log d + 2 \log \log d + O(1)$ bits.*

Interestingly enough, the adjacency labeling scheme given in Theorem 4.1 enables to give a short implicit representation (in the sense of [25]) of all forests with bounded depth. Recall that a graph G is an *induced subgraph* of a graph $\mathcal{U}$ if there exists a one-to-one (but not necessarily onto) mapping ϕ from $V(G)$ to $V(\mathcal{U})$ such that: $\forall u, v \in V(G), \quad \{u, v\} \in E(G) \iff \{\phi(u), \phi(v)\} \in E(\mathcal{U})$. Given a graph family $\mathcal{G}$, a graph $\mathcal{U}$ is *universal* for $\mathcal{G}$ if every graph in $\mathcal{G}$ is an induced subgraph of $\mathcal{U}$. Note that a variant of this notion considers the graph $\mathcal{U}$ as universal for $\mathcal{G}$ whenever every graph in $\mathcal{G}$ is a partial subgraph of $\mathcal{U}$, i.e., the existence of an edge between $\phi(u)$ and $\phi(v)$ in $E(\mathcal{U})$ does not necessarily imply the existence of the edge $\{u, v\}$. This variant enables to analyze universal graphs for infinite graph classes [33]. The notion of universality considered in this paper is somewhat more restrictive, but it enables to relate the size of a universal graph for $\mathcal{G}$ with the size of the graphs in $\mathcal{G}$. Moreover, this notion of universality precisely captures the structure of the graphs in $\mathcal{G}$. In fact, there is a tight relation between this notion and adjacency labeling schemes:

LEMMA 4.1. (S. Kannan, M. Naor, and S. Rudich [25]) *A graph family $\mathcal{G}$ has an adjacency labeling scheme with label size k if and only if there exists a universal graph for $\mathcal{G}$, with 2^k nodes.*

Combining the lemma above with Theorem 4.1, we get the corollary below.

COROLLARY 4.2. *There exists a universal graph for $\mathcal{F}(n, d)$ with $O(nd^3)$ nodes.*

Proving or disproving the existence of a universal graph with a linear number of nodes for the class of n-node trees is a central open problem in the design of informative labeling schemes.

Acknowledgments: the authors are thankful to Serge Abiteboul, Ludovic Denoyer, Sholmo Geva, and Tova Milo, for fruitful discussions about XML standard, and XML tree properties.

References

[1] S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo and T. Rauhe. Compact labeling schemes for ancestor queries. *SIAM Journal on Computing* **35**, (2006), 1295–1309.

[2] S. Abiteboul, P. Buneman and D. Suciu. Data on the Web: From Relations to Semistructured Data and XML. *Morgan Kaufmann*, (1999).

[3] Abiteboul, S., Kaplan, H., and Milo, T.: Compact labeling schemes for ancestor queries. In: Proc. 12th ACM-SIAM Symp. on Discrete Algorithms (SODA), 2001.

[4] S. Alstrup, P. Bille and T. Rauhe. Labeling Schemes for Small Distances in Trees. In: Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), 2003.

[5] S. Alstrup, C. Gavoille, H. Kaplan and T. Rauhe. Nearest Common Ancestors: A Survey and a new Distributed Algorithm. *Theory of Computing Systems* **37**, (2004), 441–456.

[6] S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43'rd annual IEEE Symp. on Foundations of Computer Science*, Nov. 2002.

[7] S. Alstrup and T. Rauhe. Improved labeling scheme for ancestor queries. In Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 947-953, 2002.

[8] Cohen, E., Kaplan, H., and Milo, T. Labeling dynamic XML trees. In Proc. 21st ACM Symp. on Principles of Database Systems (PODS), 2002.

[9] L. Denoyer. Personnal communication, 2009.

[10] L. Denoyer and P. Gallinari. The Wikipedia XML corpus. SIGIR Forum Newsletter 40(1): 64-69 (2006)

[11] A. Deutsch, M. Fernndez, D. Florescu, A. Levy and D. Suciu. A Query Language for XML. *Computer Networks* **31**, (1999), 1155-1169.

[12] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Int. Colloq. on Automata, Languages & Prog.*, LNCS 2076, pages 757–772, July 2001.

[13] C. Gavoille and A. Labourel. Distributed Relationship Schemes for Trees. In *Proc. 18th Int. Symp. on Algorithms and Computation*, pages 728-738.

[14] C. Gavoille and C. Paul. Split decomposition and distance labelling: an optimal scheme for distance hereditary graphs. In *Proc. European Conf. on Combinatorics, Graph Theory and Applications*, Sept. 2001.

[15] C. Gavoille and D. Peleg. Compact and Localized Distributed Data Structures. *J. of Distributed Computing* **16**, (2003), 111–120.

[16] C. Gavoille, D. Peleg, S. Pérennes and R. Raz. Distance labeling in graphs. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms* (SODA), pages 210–219, Jan. 2001.

[17] C. Gavoille, M. Katz, N.A. Katz, C. Paul and D. Peleg. Approximate Distance Labeling Schemes. In *9th European Symp. on Algorithms*, Aug. 2001, Aarhus, Denmark, SV-LNCS 2161, 476–488.

[18] M. Katz, N.A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing* **34** (2004),23–40.

[19] A. Korman. General Compact Labeling Schemes for Dynamic Trees. J. Distributed Computing 20(3): 179-193 (2007).

[20] A. Korman. Labeling Schemes for Vertex Connectivity. ACM Transactions on Algorithms, to appear.

[21] A. Korman. Improved Compact Routing Schemes for Dynamic Trees In *Proc. 27th Ann. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC)*, 2008.

[22] A. Korman and S. Kutten. Distributed Verification of Minimum Spanning Trees. J. Distributed Computing 20(4): 253-266 (2007).

[23] H. Kaplan and T. Milo. Short and simple labels for small distances and other functions. In *Workshop on Algorithms and Data Structures*, Aug. 2001.

[24] H. Kaplan, T. Milo and R. Shabo. A Comparison of Labeling Schemes for Ancestor Queries. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms* (SODA), Jan. 2002.

[25] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *SIAM J. on Descrete Math* **5**, (1992), 596–603.

[26] A. Korman and D. Peleg. Labeling Schemes for Weighted Dynamic Trees. J. Information and Computation 205(12): 1721-1740 (2007).

[27] A. Korman, D. Peleg, and Y. Rodeh. Labeling schemes for dynamic tree networks. *Theory of Computing Systems* **37** (2004), pp. 49-75.

[28] A. Korman, D. Peleg, and Y. Rodeh. Constructing Labeling Schemes Through Universal Matrices. *Algorithmica*, to appear.

[29] L. Mignet, D. Barbosa and P. Veltri. Studying the XML Web: Gathering Statistics from an XML Sample. *World Wide Web* **8(4)**, (2005), 413–438.

[30] I. Mlynkova, K. Toman and J. Pokorny. Statistical Analysis of Real XML Data Collections. In *Proc. 13th Int. Conf. on Management of Data*, (2006), 20 – 31.

[31] D. Peleg. Proximity-preserving labeling schemes and their applications. In *Proc. 25th Int. Workshop on Graph-Theoretic Concepts in Computer Science*, pages 30–41, June 1999.

[32] D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, volume LNCS-1893, pages 579–588. Springer-Verlag, Aug. 2000.

[33] R. Rado. Universal graphs and universal functions. Acta Arithmetica **9**:331-340, 1964.

[34] N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal* **28**, (1985), 5–8.

[35] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. of the ACM* **51**, (2004), 993–1024.

[36] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. on Parallel Algorithms and Architecture (SPAA)*, pages 1–10, Hersonissos, Crete, Greece, July 2001.

[37] W3C.Extensive markup language (XML) 1.0. http://www.w3.org/TR/REC-xml.

[38] W3C. Exensive stylesheet language (xsl) 1.0. http://www.w3.org/Style/XSL/.

[39] W3C. Xsl transformations (xslt) specification. http://www.w3.org/TR/WD-xslt