2005－AL－102（2）
2005／9／16

木写像のクラス同定

久保山　哲二†　　申　　吉浩††

† 東京大学 国際・産学共同研究センター　〒 153–8505 東京都目黒区駒場 4–6–1
†† 東京大学 先端科学技術研究センター　〒 153–8904 東京都目黒区駒場 4–6–1
E-mail: †kuboyama@ccr.u-tokyo.ac.jp　††KShin@mpeg.rcast.u-tokyo.ac.jp

概要　インターネット上の XML や HTML 文書等の半構造データの増大にともない、膨大な半構造データを効率よく比較・照合するための手法や、複数の半構造データを統合するための手法が求められている。これまでに半構造データのための様々な照合・結合手法が提案されているものの、これらの手法を統一的に記述するためのフレームワークが存在しなかった。そこで、本稿では、木の近似照合の意味論を木構造間のノード写像として定式化し、木構造間のノード写像が与えられたときに、その写像が属する近似照合のクラスを効率よく同定するアルゴリズムを示す。このアルゴリズムは、クラスの同定過程で、2 つの木構造が矛盾なく結合できるかどうかを調べ、結合可能な場合は、実際に結合木を構成する。

Class Identification of Tree Mappings

Tetsuji KUBOYAMA† and Kilho SHIN††

† Center for Collaborative Research, The University of Tokyo,
4–6–1 Komaba, Meguro-ku, Tokyo, 153–8505 Japan
†† Research Center for Advanced Science and Technology, The University of Tokyo,
4–6–1 Komaba, Meguro-ku, Tokyo, 153–8904 Japan
E-mail: †kuboyama@ccr.u-tokyo.ac.jp　††KShin@mpeg.rcast.u-tokyo.ac.jp

Abstract　With the rapid growth of semistructured data such as XML and HTML documents on the Internet, efficient methods for comparing, matching and integrating semistructured data are required. Although there have been diversity of these methods recently, no comprehensive framework has been available. We formulated a new framework of approximate tree matching in prior work. In this framework, the semantics of approximate tree matching is defined as the conditions of node-to-node correspondences between two trees. In this paper, we present two efficient algorithms for identifying a class of approximate tree matching from the node-to-node correspondences. These algorithms provide methods for merging two trees based on the node-to-node correspondences in the class identifying process if the two trees can be merged into one trees consistently.

1. Introduction

A tree structure plays a significant role in the efficient organization of information. In particular, the problem of comparing tree structures emerges across a wide range of applications in computational biology [1], image analysis [2], pattern recognition [3], natural language processing, information extraction [4] from Web pages, and many others.

Edit-based approaches provide a general framework in comparing trees, measuring similarities, finding common tree patterns, and merging trees. *Tree edit distance* [5], [6] and *alignment of trees* [7] were both introduced as natural generalizations of string edit distance [8]. It is well

known that alignment and edit are two equivalent notions in strings [9], whereas both are completely different in trees [7].

Although a dozen of tree edit methods have been proposed [10] in various fields, no comprehensive mathematical analysis of these methods had been available. In prior work [11], we have presented unifying semantics of approximate tree matching based on edit-based approaches. This semantics is defined as the conditions of node-to-node correspondences between two trees, called tree mappings. We have established the critical condition for merging two trees based on tree mappings.

In this paper, we present an efficient algorithm for identifying the tree mapping classes from tree mappings. Moreover by the effect of the algorithm, we provide a method for merging two trees.

In Section 2, we give basic notions on trees. In Section 3, we review existing classes of approximate tree matching based on the notion of tree mappings. In Section 4, we show two algorithms for identifying classes of tree mappings. In Section 5, we conclude this paper.

2. Preliminaries

In this section, we give some mathematical definitions on trees.

Trees we consider in this paper are labeled rooted trees, in which each node is labeled from a finite alphabet Σ. An *ordered tree* is a tree in which the left-to-right order among siblings is given. An *unordered tree* is a tree with no order among siblings. We refer to unordered trees as trees unless otherwise stated.

We denote by $r(T)$ the root of a tree T, and by $T(x)$ the *maximum subtree* of T rooted at a node x. An *ancestor* of a node is recursively defined as follows: an ancestor of a node is either the node itself, or an ancestor of the parent of the node.

We adopt a standard notation $<$ to denote a *strict partial order*, that is, for a non-empty finite set V, (1) $\forall x, y, z \in V\ [x < y \ \wedge \ y < z \Rightarrow x < z]$, and (2) $\forall x \in V\ [x \not< x]$. We denote by $x \leqq y$ that $x < y$ or $x = y$ for all $x, y \in V$. We say that two elements $x, y \in V$ are *comparable* if $x < y$, $x = y$ or $y < x$ holds.

Definition 1. A *rooted tree* $T = (V, <)$ is a nonempty, finite, and strict partially ordered set with the maximum element $r(T) \in V$ called the *root*, and such that $\{y \in V | x \leqq y\}$ is a totally ordered set for every $x \in V$.

Unless otherwise stated, all trees we consider in this paper are labeled, rooted and unordered trees. Although all the definitions, propositions, lemmas and theorems stated in this paper also hold for the ordered tree with no or slight modification, this paper does not state all of them.

We call the elements of V the *nodes* of T, and denote the set of all nodes in T by $V(T)$. An *ancestor* of x is a node y such that $x \leqq y$. In particular, if $x < y$, then y is called a *proper ancestor*. The *parent* of a node x is the minimum node of the proper ancestors of x in T, and denoted by $p(x)$. For a node $x \in V(T)$, we denote by $\mathrm{ch}(x)$ the set of nodes $\{y \in V(T) | y < x$ and $\nexists z \in V(T)$ such that $y < z < x\}$, and refer to the elements of $\mathrm{ch}(x)$ as the *children* of x. A *leaf* of a tree T is a minimal node in $V(T)$.

We define the notion of least common ancestor as follows.

Definition 2. For any tree $T = (V, <)$, a *common ancestor* of a set of nodes $V' \subseteqq V$ is an element $x \in V$ such that $y \leqq x$ for all $y \in V'$. A common ancestor x of V' is the *least common ancestor* of V' if, for any common ancestor x' of V', $x \leqq x'$ holds. We denote the least common ancestor of V' by $\mathrm{lca}(V')$, and $\mathrm{lca}(\{x, y\})$ by $x \frown y$.

For a node x of a tree $T = (V, <)$, we denote by $T(x)$ the subtree of T such that $T(x) = (V', <_{T'})$, $V' = \{y | y \leqq x\}$, and $\forall x, y \in V'[x < y \Leftrightarrow x <_{T(x)} y]$.

3. Classes of Approximate Tree Matching

In this section, we review a few important classes of approximate tree matching based on tree edit distance [10].

3.1 Tree Mapping

The *tree mapping* is introduced by Tai [5] as a formulation of edit distance for trees. We also refer to the *tree mapping* as *mapping* if there is no confusion. A tree mapping depicts node-to-node correspondences between two trees according to the structural similarity, or shows how nodes in one tree are preserved after transformed to the other (See Fig. 3(a)).

Definition 3 (Tai 1979 [5])**.** A *tree mapping* from a tree

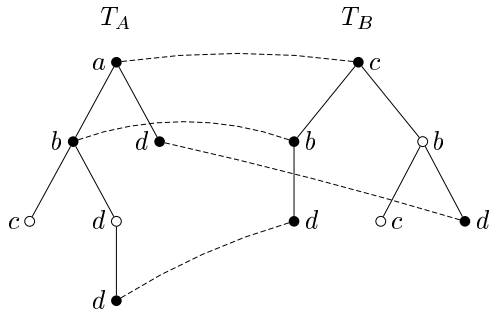


Figure 1 An example of a tree mapping

T_A to a tree T_B is a set $M \subseteqq V(T_A) \times V(T_B)$ such that, for all (x_1, x_2), $(y_1, y_2) \in M$,

(1) $x_1 \leqq y_1 \Leftrightarrow x_2 \leqq y_2$, and

(2) (only for ordered trees) x_1 is to the left of $y_1 \Leftrightarrow x_2$ is to the left of y_2.

The original definition of the tree mapping by Tai [5] includes the condition $x_1 = y_1 \Leftrightarrow x_2 = y_2$ for all (x_1, x_2), $(y_1, y_2) \in M$. We omit this condition since it is implied by the condition (1).

We denote by $\mathcal{M}(T_A, T_B)$ the set of all possible tree mappings from T_A to T_B. We simply denote $\mathcal{M}(T_A, T_B)$ by $\mathcal{M}$ if the context is clear. Figure 1 illustrates an example of a tree mapping.

We regard a set of tree mappings satisfying a condition as a class of approximate tree matching. That is, for a set of mappings $\mathcal{M}_1$ from a tree T_A to a tree T_B satisfying a condition C_1 and a set of mappings $\mathcal{M}_2$ from T_A to T_B satisfying a condition C_2, if $C_1 \Rightarrow C_2$, then $\mathcal{M}_1 \subseteqq \mathcal{M}_2$, and we say the mapping class C_1 is a subclass of C_2.

3. 2 Isolated-Subtree Mapping

Zhang *et al.* showed that the problem of finding an optimal mapping for unordered trees is NP-complete [12]. To resolve this difficulty, Zhang introduced a new mapping called a *isolated-subtree mapping* [13], which runs in polynomial time for unordered trees.

The isolated-subtree mapping originated from the *structure-preserving mapping* due to Tanaka and Tanaka [14]. Zhang gave a succinct condition of the tree mapping instead of the condition by Tanaka and Tanaka.

Definition 4 (Zhang 1996 [13]). A tree mapping M is *isolated-subtree* if the following condition holds: for all $(x_1, x_2), (y_1, y_2), (z_1, z_2) \in M$, $z_1 < x_1 \backsim y_1$ if and only if $z_2 < x_2 \backsim y_2$.

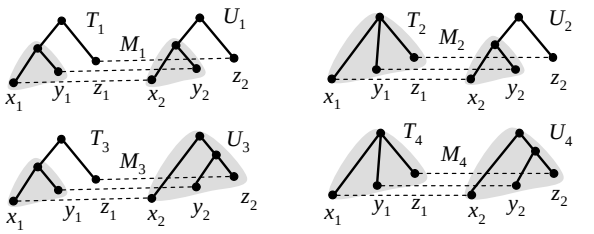


Figure 2 Examples of tree mappings: each shaded region illustrates how the subtree rooted at $x_1 \backsim y_1$ is mapped to the other by each mapping M_i ($i \in \{1, 2, 3, 4\}$). Only M_1 is isolated-subtree, and the others are not. M_1, M_2, and M_4 are alignable, and M_3 is not.

We denote by $\mathcal{M}_I$ the set of all possible mappings between two trees. This mapping was dependently proposed by a few researchers, and called constrained, structure-preserving, structure-respecting as well as isolated-subtree.

For a tree mapping M from T to U, let M_1 and M_2 be two arbitrary subsets of M. Let $X_i = \{x | (x, y) \in M_i\}$, and $Y_i = \{y | (x, y) \in M_i\}$, for $i \in \{1, 2\}$. An implication of the constrained mapping is that if $T(\mathrm{lca}(X_1))$ and $T(\mathrm{lca}(Y_1))$ are disjoint, then $U(\mathrm{lca}(X_2))$ and $U(\mathrm{lca}(Y_2))$ must be disjoint as well, and vice versa.

As shown in Fig. 2, two disjoint subtrees $T_1(x_1 \backsim y_1)$ and $T_1(z_1)$ (note that $\mathrm{lca}(\{z_1\}) = z_1$) are mapped to two disjoint subtrees $U_1(x_2 \backsim y_2)$, and $U_1(z_2)$. And the other arbitrary disjoint trees are mapped to disjoint trees by M_1. Thus, M_1 is constrained. On the other hand, two disjoint subtrees $U_2(x_2 \backsim y_2)$ and $U_2(z_2)$ are mapped to two non-disjoint subtrees $T_2(x_1 \backsim y_1)$ and $T_2(z_1)$. In fact, $T_2(x_1 \backsim y_1)$ includes $T_2(z_1)$. Thus, M_2 is not constrained.

3. 3 Alignment of Trees

The alignment of trees was introduced by Jiang *et al.* [7] as a natural extension of the alignment of strings. In contrast to the tree edit problem, the alignment of trees is viewed as the problem of finding a common supertree pattern of two trees. The definition of the alignment has been given in an operational way [7], [15] as follows.

Definition 5 (Jiang *et al.* 1995 [7]). Let T_A and T_B be two trees. An alignment of T_A and T_B is obtained by first inserting nodes labeled with the null symbol λ into T_A and T_B so that the two resulting trees T'_A and T'_B have the same structure (*i.e.*, they are identical if the labels are

ignored), and then *overlaying* T'_A on T'_B.

Figure 3 illustrates an alignment of trees. We can consider the tree mapping corresponding to the alignment of trees in Fig. 3.

We introduce an important subclass of the tree homomorphism, called embedding, which is a mapping from a tree T to a tree U such that it preserves the tree mapping condition, and $V(T) \subseteqq V(U)$.

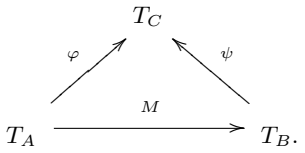
Definition 6 (Embedding)**.** Let T and U be two trees. A homomorphism $\varphi : T \to U$ is an *embedding* if the following conditions are satisfied: φ is injective, and $\forall x, y \in V(T)\ [\varphi(x) < \varphi(y) \Rightarrow x < y]$.

Definition 7. A tree mapping M from a tree T_A to a tree T_B is an *alignable* if and only if there exists a triplet (T_C, φ, ψ) such that

(1) $\varphi : T_A \to T_C$ is an embedding,

(2) $\psi : T_B \to T_C$ is an embedding, and

(3) $\varphi(x) = \psi(y)$ for all $(x, y) \in M$;



We call (T_C, φ, ψ) a *union* on M.

We showed the next theorem in [11].

Theorem 1 (Kuboyama *et al.* [11])**.** For a mapping M from a tree T_A to a tree T_B, there exists a union of T_A and T_B if and only if M satisfies the following condition: $\forall (x_1, x_2), (y_1, y_2), (z_1, z_2) \in M\ [x_1 \frown y_1 < x_1 \frown z_1 \Rightarrow y_2 \frown z_2 = x_2 \frown z_2]$.

Proposition 2. The following two conditions are equivalent.

- $\forall (x_1, x_2), (y_1, y_2), (z_1, z_2) \in M\ [x_1 \frown y_1 < x_1 \frown z_1 \Rightarrow y_2 \frown z_2 = x_2 \frown z_2]$,

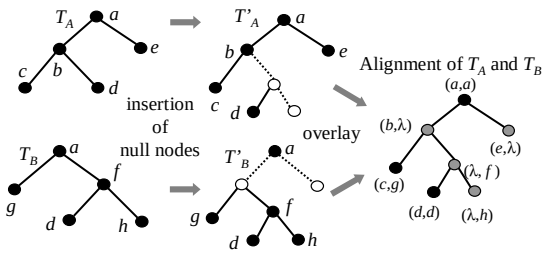


Figure 3 Example: An alignment of trees between T_A and T_B

- $\forall (x_1, x_2), (y_1, y_2), (z_1, z_2) \in M\ [x_2 \frown y_2 < x_2 \frown z_2 \Rightarrow y_1 \frown z_1 = x_1 \frown z_1]$.

We call the tree mappings satisfying the above condition as *alignable mappings*, and denote by $\mathcal{M}_A$ the set of all possible alignable mappings between two trees.

3. 4 Topological Mapping

The following mapping condition is a slightly more constrained variant of the isolated-subtree mapping.

Definition 8. A tree mapping M is *topological* if the following condition holds: for all $(x_1, x_2), (y_1, y_2), (z_1, z_2) \in M$, $x_1 \frown y_1 = x_1 \frown z_1$ if and only if $x_2 \frown y_2 = x_2 \frown z_2$.

We denote $\mathcal{M}_T$ the set of all possible topological mappings between two trees.

3. 5 Hierarchy of Tree Mapping Classes

The following relationship established among these mapping classes [11], [15].

Theorem 3. $\mathcal{M}_T \subsetneqq \mathcal{M}_I \subsetneqq \mathcal{M}_A \subsetneqq \mathcal{M}$.

4. Class Identification of Tree Mappings

In this section, we propose two algorithms for identifying the class of a tree mapping. Without loss of generality, we assume henceforth that any tree mapping includes the root-to-root correspondence.

4. 1 Decomposition of A Tree Mapping

We consider to reduce the class identification problem into easier subproblems. First, we give the definition of the decomposition of a tree as follows.

Definition 9 (Decomposition of a tree)**.** For a tree $T = (V, <)$ and a node $x \in V$, T is *decomposed* at x into $T' = (V', <_{T'})$ and $T'' = (V'', <_{T''})$ if and only if the following conditions hold.

(1) $|V'| \geqq 2$, $|V''| \geqq 2$,

(2) $V = V' \cup V''$, $V' \cap V'' = \{x\}$,

(3) $\forall x, y \in V'[x < y \Leftrightarrow x <_{T'} y]$,
$\forall x, y \in V''[x < y \Leftrightarrow x <_{T''} y]$.

We denote the decomposition of T by $T = T' \cup_x T''$ (or $T = T' \cup T''$ if the node is not specified).

Next, we give the definition of the decomposition of a tree mapping as follows.

Definition 10 (Decomposition of a tree mapping)**.** For a mapping M from a tree T_A to a tree T_B, and $(x, y) \in M$, M is *decomposed* into M', and M'' at (x, y) if and only if the following conditions hold.

(1) $|M'| \geqq 2$, $|M''| \geqq 2$, $M = M' \cup M''$,

(2) T_A is decomposed into T'_A and T''_A at x,
T_B is decomposed into T'_B and T''_B at y,

(3) $M' = M \cap (V(T'_A) \times V(T'_B))$,
$M'' = M \cap (V(T''_A) \times V(T''_B))$.

For a mapping M from a tree T_A to a tree T_B, we denote $\{x|(x, y) \in M\}$ and $\{y|(x, y) \in M\}$ $M|_1$ and $M|_2$ respectively.

Definition 11. Let M be a mapping from a tree T_A to a tree T_B. A *trimmed tree* of T with M is a tree $T'_A = (V_{T'_A}, <_{T'_A})$ such that

(1) $V_0 = M|_1$,
$V_i = \{x \smile y | x, y \in V_{i-1}\}$,
$V_{T'_A} = \bigcup_{i=0}^{\infty} V_i$,

(2) $\forall x, y \in V_{T'_A}[x < y \Leftrightarrow x <_{T'_A} y]$.

We define T'_B as well as T'_A.

Definition 12 (Simplicial mapping)**.** Let M be a mapping from a tree T_A to a tree T_B, T'_A and T'_B trimmed trees of T_A and T_B on M respectively. We decompose M, T'_A and T'_B recursively as follows until the decomposition is not applicable.

(1) decompose M, for each $(x, y) \in M$,

(2) decompose T'_A and T'_B at x and y respectively.

We denote the resulting set of mappings by $S(M) = \{M_1, \ldots, M_n\}$, and we say each $M_i \in S(M)$ for $i \in \{1, \ldots, n\}$ as a *simplicial mapping*. We denote decomposed trees of T'_A, T'_B corresponding to M_i by A_i, B_i respectively.

By the definition of simplicial mappings, it is obvious that, for trees A_i and B_i corresponding to a simplicial mapping M_i, $M_i|_1$ and $M_i|_2$ do not include any inner node (*i.e.*, a node which is neither the root nor a leaf) of A_i and B_i respectively.

We show an important property on simplicial mappings.

Proposition 4. For a mapping M and the simplicial mappings $M_1, \ldots, M_n \in S(M)$, for all $i \in \{1, \ldots, n\}$, M_i is alignable if and only if M is alignable.

Before proving this proposition, we show the next lemma which plays an important role in the proof of this proposition.

Lemma 5. $x \leqq x' \ \wedge \ y \leqq y' \Rightarrow x \smile y \leqq x' \smile y'$, and
$x \smile y < x' \smile y' \Rightarrow x \smile y \leqq x' \ \vee \ x \smile y \leqq y'$

We omit the proof since it is not hard to see.

Proof of Proposition 4. If M is alignable, it is trivial that every simplicial mapping of M is also alignable. So we show that M is alignable if every simplicial mapping of M is alignable.

For a mapping M from T_A to T_B, let M_i be a simplicial mapping of M, and A_i and B_i corresponding trees of M_i for $i \in \{1, \ldots, n\}$. Now we assume that $(a_1, b_1), (a_2, b_2), (a_3, b_3) \in M$ satisfies $a_1 \smile a_2 < a_1 \smile a_3$. Further, let $(\bar{a}, \bar{b})$ be an element of M such that $\bar{a} \in M|_1$ is the minimum node satisfying $a_i \leqq \bar{a}$ for any $i \in \{1, 2, 3\}$. Note that $\bar{b} \in M|_2$ is also the minimum node such that $b_i \leqq \bar{b}$ for any $i \in \{1, 2, 3\}$.

Now for $d_i = |\{(a, b) \in M | a_i < a \leqq \bar{a}\}|$, we prove $b_1 \smile b_3 = b_2 \smile b_3$ by induction on $d = \max\{d_1, d_2, d_3\}$.

It is trivial if $d = 1$. Therefore we assume $d \geqq 2$. For all $i \in \{1, 2, 3\}$ such that $d = d_i$, let (a'_i, b'_i) be an element of M such that $a'_i < \bar{a}$ and $b_i < b'_i < \bar{b}$. For all $i \in \{1, 2, 3\}$ such that $d < d_i$, let $(a'_i, b'_i) = (a_i, b_i)$.

First, we claim that $a'_1 \smile a'_2 < a'_1 \smile a'_3$. By Lemma 5, either $a_1 \smile a_2 = a'_1 \smile a'_2$ or $a_1 \smile a_2 \in \{a'_1, a'_2\}$ holds. On the other hand, $a'_i < a_1 \smile a_3 \leqq a'_1 \smile a'_3$ holds by the choice of a_i for some $i \in \{1, 2, 3\}$. It follows that $a'_1 \smile a'_2 < a'_1 \smile a'_3$. Therefore, the hypothesis of the induction implies $b'_1 \smile b'_3 = b'_2 \smile b'_3$.

Next, we claim $b_i \smile b_3 = b'_i \smile b'_3$ for any $i \in \{1, 2\}$. By Lemma 5, if $b_i \smile b_3 < b'_i \smile b'_3$ holds, then $b'_i \smile b'_3 \in \{b'_i, b'_3\}$ holds. Thus, either $\bar{b} \leqq b'_i$ or $\bar{b} \leqq b'_3$ holds, a contradiction to the choice of b_i and $\bar{b}$. Then, we have the claim $b_i \smile b_3 = b'_i \smile b'_3$ for any $i \in \{1, 2\}$.

Finally, we conclude that $b_1 \smile b_3 = b'_1 \smile b'_3 = b'_2 \smile b'_3 = b_2 \smile b_3$. □

Intuitively, we can see that if a union of A_i and B_i exists on every simplicial mapping M_i, a union of T_A and T_B also exists on M.

Proposition 6. For a mapping M and the simplicial

mappings $M_1, \ldots, M_n \in S(M)$, for all $i \in \{1, \ldots, n\}$, M_i is isolated-subtree if and only if M is isolated-subtree.

Proof. We assume that $a_1 < a_2 \frown a_3$ holds for $a_1, a_2, a_3 \in M|_1$, and show $b_1 < b_2 \frown b_3$.

In the rest of proof, we use the following notations.

- $\bar{a} \in M|_1$ is the minimum node of T_A such that $a_2 \frown a_3 \leqq \bar{a}$. When $\bar{b} \in M|_2$ is the node of T_B such that $(\bar{a}, \bar{b}) \in M$, $\bar{b}$ is also the minimum node such that $b_2 \frown b_3 \leqq \bar{b}$.
- For any $i \in \{1, 2, 3\}$, $a'_i \in M|_1$ is the maximum node of T_A such that $a_i \leqq a'_i < \bar{a}$. When $b'_i \in M|_2$ is the node of T_B such that $(a'_i, b'_i) \in M$, b'_i is also the maximum node such that $b_i \leqq b'_i < \bar{b}$.

We have $a'_1 < a'_2 \frown a'_3$. In fact, if we assume $a'_1 \geqq a'_2 \frown a'_3$, then it follows that $a'_1 \geqq \bar{a}$ since it contradicts the definition of a'_1. Note that both a'_1 and $a'_2 \frown a'_3$ are comparable because both nodes are ancestors of a_1.

Therefore, we have $b'_1 < b'_2 \frown b'_3$ since every simplicial mapping is isolated-subtree. Furthermore, by Lemma 5, $b_2 \frown b_3 = b'_2 \frown b'_3$ holds since otherwise we have either $\bar{b} \leqq b'_2$ or $\bar{b} \leqq b'_3$, a contradiction to the choice of $\bar{b}$.

Thus, we have $b_1 \leqq b'_1 < b'_2 \frown b'_3 = b_2 \frown b_3$. □

4.2 Top-Down Algorithm

For a mapping M from T_A to T_B and the simplicial mappings $M_1, \ldots, M_n \in S(M)$, by Proposition 4 and 6 it suffices to check the class of M_i for each $i \in 1, \ldots, n$. So, we show a bottom-up algorithm for identifying the class of M by assuming that M is a simplicial mapping from T_A to T_B in the rest of this paper.

Let M be a simplicial mapping $\{(r(T_A), r(T_B)), (a_1, b_1), \ldots, (a_n, b_n)\}$, where a_i and b_i are all leaves of T_A and T_B respectively. This algorithm which we show in this section is available for unrooted trees as well as rooted trees. In this case, a simplicial mapping is represented as $\{(a_1, b_1), \ldots, (a_n, b_n)\}$, where a_i and b_i are all leaves of T_A and T_B respectively.

The algorithm performs two scans. One is the preliminary scan, which is optional and performed only when it is necessary, and the other is the class identification scan.

a) Preliminary Scan. For any $x \in V(T_A) \cup V(T_B)$, by Σ_x we denote $\{i | a_i \leqq x\}$ if $x \in V(T_A)$, and $\{i | b_i \leqq y\}$ if $x \in V(T_B)$. Then, for any $i \in \Sigma_x$ and $j \in \Sigma_{p(y)} \setminus \Sigma_x$ (we assume $x \in V(T_A)$), the node $p(x)$ is identical to $a_i \frown s_j$.

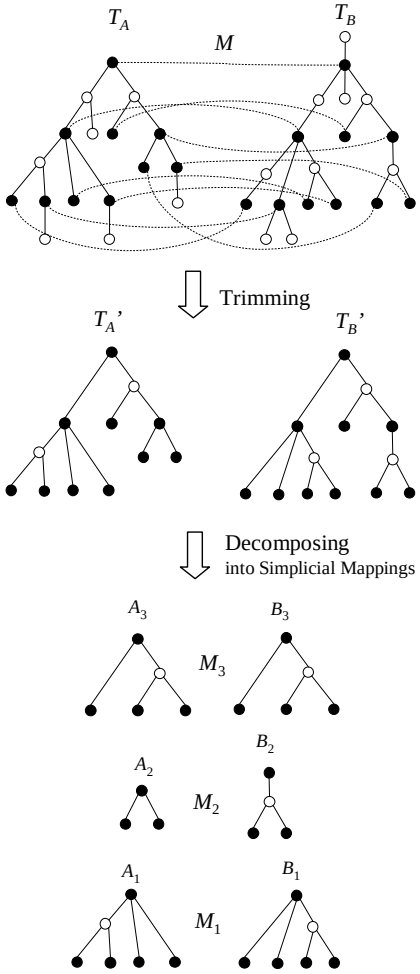


Figure 4 An example of decomposition of a tree, and simplicial mappings

The time complexity of calculating Σ_x is linear to the size of the trees. In fact, the following algorithm visits a node of $V(T_A)$ no more than twice to calculate Σ_x for all $x \in V(T_A)$.

(1) For each $a_i \in M|_1$,
  (a) Set $x \leftarrow a_i$.
  (b) Set $\Sigma_x \leftarrow \Sigma_x \cup \{i\}$.
  (c) If $x \neq r(T_A)$ and $\Sigma_{p(x)} \neq \emptyset$, then set $x \leftarrow p(x)$ and go to (2).

(2) Scan all the nodes $x \in V(T_A)$ from the leaves to the root and set $\Sigma_x \leftarrow \bigcup_{y \in \mathrm{ch}(x)} \Sigma_y$.

b) Class Identification Scan. Let $x \in V(T_A)$ be a maximal node such that $\Sigma_x \subsetneq \{1, 2, \ldots, n\}$ and $\Sigma_x \neq \emptyset$. Further, let $y \in V(T_B)$ be the minimum node such that $\Sigma_y = \{1, 2, \ldots, n\}$.

We employ the following notations.

- $\Delta = \{1, 2, \ldots, n\} \setminus \Sigma_x$.

- $M_x = \{(a_i, b_i) | i \in \Sigma_x\}$.
- $M_\Delta = \{(a_i, b_i) | i \in \Delta\}$.

Given a simplicial mapping M, the following procedure identifies M into the classes of the alignable, isolated-subtree and topological mappings and the other.

(1) The flags of ALIGNABLE, ISOLATED-SUBTREE, and TOPOLOGICAL are initialized by `true`.

(2) Identify the class of M_x. The preliminary scan is skipped, and the result is set to the all flags.

(3) Identify the class of M_Δ. The preliminary scan is skipped, and the result is set to the all flags.

(4) Set all the flags as `false` when $\Sigma_z \cap \Sigma_x = \emptyset$ and $\Sigma_z \cap \Delta = \emptyset$ for some child $z \in \mathrm{ch}(y)$.

(5) Set ISOLATED-SUBTREE and TOPOLOGICAL as `false` when either of the following conditions holds:

(a) $\Sigma_z \subsetneq \Sigma_x$, and $\Sigma_z \neq \emptyset$ for some $z \in \mathrm{ch}(y)$.

(b) $\Sigma_z \neq \Sigma_u$ and $\Sigma_z \cap \Sigma_u \cap \Delta \neq \emptyset$ for some $z \in \mathrm{ch}(y)$ and $u \in \mathrm{ch}(x)$.

(6) If M is a rooted simplicial tree mapping, set TOPOLOGICAL←`false` when either of the following conditions holds:

(a) There exists $z \in \mathrm{ch}(r(T_A))$ such that $\Sigma_z = \{1, \ldots, n\}$ and $y = r(T_B)$.

(b) There does not exist $z \in \mathrm{ch}(r(T_A))$ such that $\Sigma_z = \{1, \ldots, n\}$ and $y < r(T_B)$.

We omit the proof of the correctness of this top-down algorithm due to the space limitation. This algorithm runs in polynomial time to the sizes of trees.

4.3 Bottom-Up Algorithm

A node whose children are all leaves is called *cherry*. Consider a mapping $M^- = M \setminus \{(r(T_A), r(T_B))\} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ such that M is a simplicial mapping from T_A to T_B with uniquely numbering each element of M, and we henceforth denote M^- simply by M. For $L \subseteqq V(T_A) \cup V(T_B)$, we denote $\{(a, b) \in M | a \in L \ \vee \ b \in L\}$ by $M[L]$. Moreover, let $M(L)$ denote $\{b | (a, b) \in M[L]\}$ if $L \subseteqq V(T_A)$, and let $M(L)$ denote $\{a | (a, b) \in M[L]\}$ if $L \subseteqq V(T_B)$. We denote by $\mathrm{ch}_L(x)$ the set of all leaves in $\mathrm{ch}(x)$.

Now we can show a bottom-up algorithm for class identification as follows:

(1) The flags of ALIGNABLE, ISOLATED-SUBTREE, and TOPOLOGICAL are initialized by `true`.

(2) Select a cherry x with the minimum number of leaves from $V(T_A) \cup V(T_B)$.

(3) Set $x' \leftarrow \mathrm{lca}(M(\mathrm{ch}(x)))$.

(a) If x' is also a cherry and $M[\mathrm{ch}(x)] = M[\mathrm{ch}(x')]$ holds, then remove all the nodes in $\mathrm{ch}(x)$ and $\mathrm{ch}(x')$, and let $M \leftarrow M \setminus M[\mathrm{ch}(x)]$. Moreover, let $M \leftarrow M \cup \{(x, x')\}$ if $x \in V(T_A)$, and let $M \leftarrow M \cup \{(x', x)\}$ if $x \in V(T_B)$.

(b) If $M[\mathrm{ch}(x)] \subseteqq M[\mathrm{ch}_L(x')]$ holds, then remove all the nodes in $\mathrm{ch}(x)$ and $M(\mathrm{ch}(x))$, add a new node x^* as a child of x', and let $M \leftarrow M \setminus M[\mathrm{ch}(x)]$. Moreover, let $M \leftarrow M \cup \{(x, x^*)\}$ if $x \in V(T_A)$, and let $M \leftarrow M \cup \{(x^*, x)\}$ if $x \in V(T_B)$. Set ISOLATED-SUBTREE←`false`, and TOPOLOGICAL←`false`.

(c) If $M[\mathrm{ch}(x)] \supsetneqq M[\mathrm{ch}_L(x')]$ holds, then set all flags as `false`.

(4) Go to (2) if $|M| \geq 2$ holds.

(5) Set TOPOLOGICAL←`false` unless both T_A and T_B are truncated into trivial trees.

This algorithm runs in polynomial time to the sizes of trees, and in linear time if the maximum number of children is bounded.

4.4 Correctness of Bottom-Up Algorithm

By $T\langle L \rangle$, we denote a tree obtained by replacing a set of leaves L in a tree T with a new node. For a mapping M from T_A to T_B, we denote a mapping from $T_A\langle L \rangle$ to $T_B\langle M(L) \rangle$ as follows:

$$M\langle L \rangle = M \setminus M[L] \cup \{(a^*, b^*)\},$$

where we assume that (a^*, b^*) is an element of M added in the bottom-up algorithm.

We have the following lemma from Lemma 5.

Lemma 7. For a node x in a tree T, let $y \in \mathrm{ch}_L(x)$ and $z \notin \mathrm{ch}_L(x)$ in the tree $T\langle \mathrm{ch}_L(x) \rangle$. Then, $y \backsim z = y^* \backsim z$ holds, where y^* is a node replacing $\mathrm{ch}_L(x)$.

Proof. We have $x \leqq y \backsim z$ since y is a leaf, and a child of x. It follow from Lemma 5 that $y \backsim z = y^* \backsim z$ holds. □

Now we ready to prove the correctness of the bottom-up algorithm.

Theorem 8. M is an alignable mapping from a tree T_A to a tree T_B if and only if, for a cherry $x \in$

$V(T_A) \cup V(T_B)$ with the minimum number of leaves, $M[\mathrm{ch}(x)] \subseteqq M[\mathrm{ch}_L(\mathrm{lca}(M(\mathrm{ch}(x))))]$ holds and $M\langle \mathrm{ch}(x)\rangle$ is an alignable mapping.

Proof. Now we assume without loss of generality that $a \in V(T_A)$ is a cherry with the minimum number of leaves in $V(T_A) \cup V(T_B)$ and $|\mathrm{ch}(a)| \geqq 2$. Let $a_1 \in \mathrm{ch}(a)$, and $b \in p(b_1)$.

It suffices to show that $M[\mathrm{ch}(a)] \subseteqq M[\mathrm{ch}_L(b)]$ holds if M is alignable. The other part of the statement is obvious by Lemma 7.

First, we assume that $L_B(b) = \{b_i | b_i < b \wedge b_i \text{ is a leaf}\}$ holds. Then, we show that $M[\mathrm{ch}(a)] \subseteqq M[L_B(b)]$ holds by deriving a contradiction under the assumption $b_i \not< b$ for $b_i \in M(\mathrm{ch}(a))$.

$M[L_B(b)] \setminus M[\mathrm{ch}(a)] \neq \emptyset$ holds due to the minimality of $|M[\mathrm{ch}(a)]|$. Hence, for $(a_j, b_j) \in M$ such that $b_j \in L_B(b) \setminus M(\mathrm{ch}(a))$, we have $a_1 \smile a_i < a_1 \smile a_j$ and $b_1 \smile b_i > b = b_1 \smile b_j$. This contradicts that M is alignable.

It is straightforward from $M[\mathrm{ch}(a)] \subseteqq M[L_B(b)]$ that $M[\mathrm{ch}(a)] \subseteqq M[\mathrm{ch}_L(b)]$ holds. □

As in the proof of Theorem 8, we can show the following theorem.

Theorem 9. *M is an isolated-subtree mapping from a tree T_A to a tree T_B if and only if, for a cherry $x \in V(T_A) \cup V(T_B)$, $M\langle \mathrm{ch}(x)\rangle$ is an isolated-subtree mapping, and $M[\mathrm{ch}(x)] = M[\mathrm{ch}_L(\mathrm{lca}(M(\mathrm{ch}(x))))]$.*

These two theorems guarantee the correctness of the bottom-up algorithm.

5. Conclusion

In this paper, we have presented two efficient algorithms for identifying the class of given tree mappings. By these algorithms, we can determine whether two trees can be merged into one tree based on a tree mapping. Moreover, these algorithms provide methods for merging two trees in a top-down and a bottom-up manner. That is, we have provided a theoretical framework for merging two semistructured data such as XML and HTML documents. As future work, we plan to apply our framework to an actual data set of semistructured data for merging similar documents.

References

[1] Y. Sakakibara. Pair hidden markov models on tree structures. *Bioinformatics*, 19:232–240, 2003.

[2] A. Torsello and E. R. Hancock. Matching and embedding through edit-union of trees. *LNCS*, 2352:822–836, 2002. ECCV 2002.

[3] P. Ferraro and C. Godin. A distance measure between plant architectures. *Annals of Forest Science*, 57:445–461, 2000.

[4] A. Hogue and D. Karger. Thresher: Automating the unwrapping of semantic content from the world wide web. In *Proc. of WWW 2005*, pages 86–95, 2005.

[5] K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, July 1979.

[6] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, December 1989.

[7] T. Jiang, L. Wang, and K. Zhang. Alignment of trees — an alternative to tree edit. *Theoretical Computer Science*, 143:137–148, 1995.

[8] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[9] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[10] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239, 2005.

[11] T. Kuboyama, S. Kilho, T. Miyahara, and H. Yasuda. A theoretical analysis of alignment and edit problems for trees. In *Proc. of Theoretical Computer Science, The 9th Italian Conference, LNCS 3701*, pages 323–337, 2005.

[12] K. Zhang, R. Statman, and D. Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992.

[13] K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15:205–222, 1996.

[14] E. Tanaka and K. Tanaka. The tree-to-tree editing problem. *IJPRAI*, 2:221–240, 1988.

[15] J.T.-L. Wang and K. Zhang. Finding similar consensus between trees: an algorithm and a distance hierarchy. *Pattern Recognition*, 34:127–137, 2001.