NUMERIC SIMILARITY AND DISSIMULARITY MEASURES BETWEEN TWO TREES

B. J. Oommen, K. Zhang and W. Lee

TR-203, FEBRUARY 1992
(REVISED JULY 1993)

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

NUMERICAL SIMILARITY AND DISSIMILARITY MEASURES BETWEEN TWO TREES+

B. J. Oommen* , K. Zhang and W. Lee***

ABSTRACT

Quantifying the measure of dissimilarity between two trees is a problem of intrinsic importance in the study of algorithms and data structures. But it is also of paramount importance in the fields of structural and syntactic pattern recognition. In this paper we define and formulate an abstract measure of comparison, $\Omega(T_1, T_2)$, between two trees T_1 and T_2. This measure is presented in terms of a set of elementary inter-symbol measures $\omega(.,.)$ and two abstract operators $\oplus$ and $\otimes$. By appropriately choosing the concrete values for these two operators and for $\omega(.,.)$, this measure can be used to define the edit distance between two trees and the Size of their Largest Common Sub-Tree (SLCsT). Additionally, a special case of this measure can also be used to quantify $\mathrm{Prob}(T_2|T_1)$, the probability of receiving T_2 given that T_1 was transmitted across a channel causing independent substitution and deletion errors. Finally, the same abstract measure can be used to quantify the ***a posteriori*** probability of T_1 being the transmitted tree given that T_2 is the received tree containing independent substitution, insertion and deletion errors. Apart from the definition and formulation of the abstract measure $\Omega(T_1, T_2)$, we have derived the recursive properties of the measure and presented an iterative dynamic programming scheme to compute it. The analysis of the time and space complexities of the algorithm are also included. If for a tree T, Span(T) is defined as the Min{Depth(T), Leaves(T)}, and if the operations $\oplus$ and $\otimes$ are assumed to take unit time, the time and space complexities of this algorithm are :

Time : $O(|T_1| * |T_2| * \mathrm{Span}(T_1) * \mathrm{Span}(T_2))$

Space : $O(|T_1| * |T_2|)$.

+ A preliminary version of some of the results of this paper was presented at the International Symposium on Computer and Information Sciences, Antalya, Turkey in November 1992. The work of the first and second authors was supported by the Natural Sciences and Engineering Coucil of Canada.

* School of Computer Science, Carleton University, Ottawa ; CANADA : K1S 5B6.

** Department of Computer Science, University of Western Ontario, London, CANADA : N6A 5B7.

I. INTRODUCTION

Trees, graphs, and webs are typically considered as multi-dimensional generalization of strings. Among these different structures trees are considered to be the most important "non-linear" structures in computer science, and the tree-editing problem has been studied since 1976. Apart from its purely theoretical significance, the problem has also paramount significance in structural and syntactic pattern recognition. The reason for this is, of course, quite straightforward. When the pattern to be recognized is inherently a "two-dimensional" structure, it is often the case that the pattern cannot be adequately represented by a one-dimensional approximation such as a string or even as a circular string. By representing the pattern as a tree and by utilizing tree comparison algorithms one can, generally speaking, achieve excellent recognition strategies. Indeed, such schemes have been utilized in pattern recognition in areas such as clustering [Lu79, Lu84], waveform correlation [CL85] and the classification of biological macromolecules [SZ90]. Furthermore, researchers have also used tree comparison algorithms in the automatic error recovery and correction of programming languages [Ta79].

Similar to the string-to-string editing problem [KO83, KO84, WF74], the tree-to-tree editing problem concerns the determination of the distance between two trees as measured by the minimum cost sequence of edit operations. Typically, the edit sequence considered, includes the substitution, insertion, and deletion of nodes needed to transform one tree into the other. Possible applications of the tree-to-tree editing problem can be found in the theory of amino-acid sequence comparison, pattern recognition, and in the parsing of sentences from a grammar. As an example, consider the secondary structure comparisons of RNA in the study of macromolecules. It is well known that the secondary structure of RNA is a single strand of nucleotides which folds back onto itself into a shape that is topologically a tree [SK83, SZ90, ZS89]. This strand influences the translation rates from RNA to proteins. Thus comparisons among these secondary structures are necessary to understand the comparative functionality of different RNAs.

Unlike the string-editing problem which has been well-developed, only few results have been published concerning the tree-editing problem. In 1977, Selkow [Se77] presented a tree editing algorithm in which insertions and deletions were only restricted to the leaves. In 1979, Tai [Ta79] presented another algorithm in which insertions and deletions could take place at any node within the tree except the root. The algorithm of Lu [Lu79], on the other hand, did not solve this problem for trees of more than two levels. The best known algorithm for solving the general tree-editing problem is the one due to Zhang and Shasha [ZS89].

For a given pair of strings, there are many metrics which quantify the similarity and dissimilarity between them. An excellent treatise on the problems related to the editing of strings is the one due to Sankoff *et. al.* [SK83]. Their Generalized Levenshtein Distance (GLD) (a partial list of references using this is [Al67], [HD80], [K083], [LW75], [MP80], [WF74] and [WC76]), the Length of their Longest Common Subsequence (LLCS) (a partial list of references using this is [AHU76], [Hi75, 77, 78], [HS77], [KO83] and [KT82]), the Length of their Shortest Common Supersequence (LSCS) ([KO83], [Ma78]), and their transmission-reception probabilities ([KO82], [KO83], [KO84]) are examples of these numeric indices. Others, like the set of their Longest Common Subsequences, the set of their Shortest Common Supersequences, and the set of their Shuffles are the examples of various nonnumeric quantities. In 1983, Kashyap and Oommen [KO83] presented a common mathematical framework by which ***all*** these quantities can be related. In [KO83], they defined an abstract measure between two strings X and Y, denoted by D(X, Y). The latter was defined in terms of two abstract operators, and a binary function, d(.,.), whose arguments are essentially symbols of the alphabet under consideration and the null symbol. Depending on various concrete operators used for the abstract operators and the specific function used for d(.,.), all the numeric and nonnumeric quantities discussed above can be seen to be particular cases of D(X, Y).

As opposed to the case of strings, to the best of our knowledge, the literature primarily reports only one numeric dissimilarity measure relating two trees. This measure is indeed the "distance" between the trees as measured by the minimum cost sequence of edit operations. Typically, the edit sequence considered includes the substitution, insertion, and deletion of nodes needed to transform one tree into the other. In this paper, we define a few more new numeric indices which measure the similarity between two trees. These include the Size of their Largest Common Sub-Tree (SLCsT), and the transmission-receptive probabilities of the trees when the transmission is done across channels with a variety of properties. Indeed, more generally, in this paper we will define entire families of numeric measures involving the trees in question.

The central issue in this paper is the definition and formulation of an abstract measure of comparison, $\Omega(T_1, T_2)$, between two trees T_1 and T_2. This measure will be presented in terms of a set of elementary inter-symbol measures $\omega(.,.)$ and two abstract operators $\oplus$ and $\otimes$. By appropriately choosing the concrete values for these two operators and for $\omega(.,.)$, the edit distance between the trees and their SLCsT can be obtained as special cases. The quantity $\text{Prob}(T_2|T_1)$, the probability of receiving T_2 given that T_1 was transmitted across a channel causing independent substitution and deletion errors, can also be seen to be another special case of $\Omega(T_1, T_2)$. Finally, the same abstract measure can be used to quantify the probability of T_1 being the transmitted tree given that T_2 is the received tree containing independent substitution, insertion and deletion errors and that the transmitted tree is an element from a finite dictionary.

Apart from the definition and formulation of the abstract measure $\Omega(T_1, T_2)$, we shall also derive the recursive properties of the measure whence an iterative dynamic programming scheme to compute $\Omega(T_1, T_2)$ will be presented. Thus, the same algorithm with the appropriate choice of the operators and of $\omega(.,.)$ can be used to compute all of the above measures.

The original novel concept of utilizing generalized operators to compute numeric and non-numeric quantities on a graph was first proposed by Aho, Hopcroft and Ullman in their excellent book [AHU74]. They did this by introducing the "closed semi-ring" model of computation. From a naive perspective it may appear that the results of this paper essentially utilize these concepts in the setting of the tree-editing problem. This is ***definitely*** not the case because, although the pioneering work in [AHU74] explains how the semi-ring can be defined, it does not consider the possibility that the topological structure of the underlying graph may prohibit the abstractions of the computations. Indeed, our results do not merely have pedagogical significance, since we show that the general abstraction of the tree-editing problem is not possible using the closed semi-ring model because tree transformations require the simultaneous maintenance of the parent and sibling properties of the respective trees. We shall clarify this in a subsequent section.

II. NOTATIONS AND DEFINITIONS

2.1 Notation

Let $\mathcal{N}$ be an alphabet and $\mathcal{N}^*$ be the set of trees whose nodes are elements of $\mathcal{N}$. Let μ be the null tree, which is distinct from λ, the null node. Let $\tilde{\mathcal{N}} = \mathcal{N} \cup \{\lambda\}$. $\tilde{\mathcal{N}}$ is referred to as the Appended Alphabet. A tree $T \in \mathcal{N}^*$ is said to be of size $|T| = M$ if it contains M nodes.

A tree will be represented in terms of the postorder numbering of its nodes. The advantages of this ordering are catalogued in [ZS89]. Let T[i] be the i^{th} node in the tree according to the left-to-right postorder numbering, and let $\delta(i)$ represent the postorder number of the leftmost leaf descendant of the subtree rooted at T[i]. Note that when T[i] is a leaf, $\delta(i) = i$.

T[i..j] represents the postorder forest induced by nodes T[i] to T[j] inclusive, of tree T. $T[\delta(i)..i]$ will be referred to as Tree(i). Size(i) is the number of nodes in Tree(i). An example of these terms is shown pictorially in Figure I.

*** * * * *   Insert Figure I Here   * * * * ***

Finally, the father of a node i is denoted as f(i). If we define $f^0(i) = i$, the node $f^k(i)$ can be recursively defined as $f^k(i) = f(f^{k-1}(i))$. We define the set of ancestors of i as :

$\text{Anc}(i) = \{f^k(i) \mid 0 \le k \le \text{Depth}(i)\}$.

In the body of the paper we will use Figure I to explain various notational details.

2.2 Elementary Edit Operations

An edit operation on a tree is either an insertion, a deletion or a substitution of one node by another. In terms of notation, an edit operation is represented symbolically as : $x \rightarrow y$ where x and y can either be a node value or λ, the null node. $x = \lambda$ and $y \neq \lambda$ represents an insertion; $x \neq \lambda$ and $y = \lambda$ represents a deletion; and $x \neq \lambda$ and $y \neq \lambda$ represents a substitution. Note that the case of $x = \lambda$ and $y = \lambda$ has not been defined -- it is not needed. The formal definitions of these follow.

(i) An insertion of node x into tree T.

Node x will be inserted as a son of some node u of T. It may either be inserted with no sons or take as sons any subsequence of the sons of u. Formally, if u has sons $u_1, u_2, \dots, u_k$, then for some $0 \leq i \leq j \leq k$, node u in the resulting tree will have sons $u_1, \dots, u_i, x, u_j, \dots, u_k$, and node x will have no sons if $j = i+1$, or else have sons $u_{i+1}, \dots, u_{j-1}$. An example of insertion is shown in Figure II.

* * * * *	**Insert Figure II Here**	* * * * *

(ii) A deletion of node y from a tree T.

If node y has sons $y_1, y_2, \dots, y_k$ and node u, the father of y, has sons $u_1, u_2, \dots, u_j$ with $u_i = y$, then node u in the resulting tree obtained by the deletion will have sons $u_1, u_2, \dots, u_{i-1}, y_1, y_2, \dots, y_k, u_{i+1}, \dots, u_j$. An example of deletion is shown in the Figure III.

* * * * *	**Insert Figure III Here**	* * * * *

(iii) Substitution of node x by node y in T.

In this case, node y in the resulting tree will have the same father and sons as node x in the original tree. An example of substitution is shown in Figure IV.

* * * * *	**Insert Figure IV Here**	* * * * *

2.3 Common Sub-Trees of Trees

A tree $T_1 \in \mathcal{N}^*$ is a *Sub-Tree* of a tree $T_2 \in \mathcal{N}^*$ if it can be obtained from T_2 by deleting zero or more nodes from T_2. It is a *Common Sub-Tree* of two trees T_2 and T_3 if it is a Sub-Tree of both T_2 and T_3, and it is their *Largest Common Sub-Tree (LCsT)* if and only if it is a common sub-tree of both that is among the largest of the common sub-trees. The size of the LCsT is referred to as the *Size of their Largest Common Sub-Tree (SLCsT)*. Examples of these are shown in Figure V.

* * * * *	**Insert Figure V Here**	* * * * *

III. AN ABSTRACT MEASURE INVOLVING TWO TREES

We consider a functional $\Omega(T_1, T_2)$, between two trees T_1 and T_2, $T_1, T_2 \in \mathcal{N}^*$, induced by a system $(\mathcal{N}, \tau, \omega)$, where $\mathcal{N}$ is the alphabet set under consideration, τ is an algebraic structure described below and ω is a binary function over $\tilde{\mathcal{N}} \times \tilde{\mathcal{N}}$. τ and ω are defined as follows.

3.1 The Abstract Structure, τ, and the Set of Elementary Measures, ω

The structure τ is defined by the 5-tuple $(T, \oplus, \otimes, \theta, I)$, where :

i) $(T,\oplus,\theta)$ is a commutative monoid [KO83], i.e., T is a finite or infinite set, $\oplus$ is an associative and commutative operator defined from $T \times T$ to T, and T is closed with respect to $\oplus$. Also, $\theta \in T$ is the identity for $\oplus$.

ii) $(T,\otimes,I)$ is a commutative monoid with $I \in T$ as the identity for $\otimes$.

iii) $\otimes$ distributes over $\oplus$ from both sides.

The iterative use of $\oplus$ and $\otimes$ are denoted by $\textcircled{\scriptsize\Sigma}$ and $\textcircled{\scriptsize\Pi}$ respectively, defined below for all $h_i \in T$.

$$\underset{i=1}{\overset{n}{\textcircled{\scriptsize\Sigma}}} h_i = h_1 \oplus h_2 \oplus \dots \oplus h_n$$

$$\underset{i=1}{\overset{n}{\textcircled{\scriptsize\Pi}}} h_i = h_1 \otimes h_2 \otimes h_3 \dots \otimes h_n.$$

EXAMPLE III : Consider the structure $\tau_1 = (R_p', \text{MIN}, +, \infty, 0)$ where :

i) R_p' is the set consisting of the non-negative real numbers and ∞.

ii) MIN is the binary operator defined for all $p, q \in R_p'$ by :

$$\text{MIN}(p, q) = \begin{cases} p & \text{if } p \le q; \\ q & \text{otherwise.} \end{cases}$$

and iii) + represents arithmetic addition.

τ_1 is a well-defined structure since $(R_p', \text{MIN}, \infty)$ is a commutative monoid, $(R_p', +, 0)$ is a monoid, and + distributes over MIN from both sides. ***

$\omega(.,.)$ is a function whose arguments are a pair of nodes belonging to $\tilde{\mathcal{N}}$, the Appended Alphabet, and whose range is T, the set defined in τ. $\omega(\lambda, \lambda)$ is undefined since it is not needed. The elementary measure $\omega(x, y)$ can be interpreted as the measure associated with transforming "x" to "y", for $x, y \in \tilde{\mathcal{N}}$. In other words, $\omega(x, y)$ will represent the "cost" (weight or advantage) of substituting node x by node y if $x \neq \lambda \neq y$. Similarly, $x \neq \lambda$, $y = \lambda$ and $x = \lambda$, $y \neq \lambda$ will represent the cost of deletion and insertion of nodes x and y respectively.

3.2 Mappings, Extended Mappings and the set $\Gamma_{(T_1,T_2)}$

A Mapping is a description of how a sequence of edit operations transforms T_1 into T_2. Informally, we can describe a mapping as follows :

(i) Lines connecting $T_1[i]$ and $T_2[j]$ correspond to substituting $T_1[i]$ by $T_2[j]$.

(ii) Nodes in T_1 not touched by any line are to be deleted.

(iii) Nodes in T_2 not touched by any line are to be inserted.

Formally, a mapping is a triple (M,T_1,T_2), where M is any set of pairs of integers (i,j) satisfying :

(i) $1 \leq i \leq |T_1|$, $1 \leq j \leq |T_2|$;

(ii) For any pair of (i_1,j_1) and (i_2,j_2) in M,

(a) $i_1 = i_2$ if and only if $j_1 = j_2$ (one-to-one).

(b) $T_1[i_1]$ is to the left of $T_1[i_2]$ if and only if $T_2[j_1]$ is to the left of $T_2[j_2]$. This is referred to as the Sibling Property.

(c) $T_1[i_1]$ is an ancestor of $T_1[i_2]$ if and only if $T_2[j_1]$ is an ancestor of $T_2[j_2]$. This is referred to as the Ancestor Property.

A pictorial representation of a mapping is given in Figure VI for a sample pair of trees.

*** * * * *   Insert Figure VI Here   * * * * ***

Whenever there is no ambiguity we will use M to represent the triple (M,T_1,T_2), the mapping from T_1 to T_2. Let I, J be sets of nodes in T_1 and T_2, respectively, not touched by any lines in M. Then we can define C(M) as the abstract index associated with M as follows :

$$C(M) = \underset{(i,j)\in M}{\textcircled{\sqcap}} \omega(T_1[i], T_2[j]) \otimes \underset{(i\in I)}{\textcircled{\sqcap}} \omega(T_1[i],\lambda) \otimes \underset{(j\in J)}{\textcircled{\sqcap}} \omega(\lambda,T_2[j]).$$

For every pair (T_1, T_2), $T_1,T_2 \in \mathcal{N}^*$, the finite set $\Gamma_{(T_1,T_2)}$ is defined as follows. Let M be a mapping from T_1 to T_2 defined as above. Let I, J be sets of nodes in T_1 and T_2, respectively, not touched by any lines in M. Then, with respect to M, we define the *Extended Mapping*, denoted by M', between T_1 and T_2 as :

$$M' = \{(T_1[x], T_2[y]) \mid (x, y) \in M\} \cup \{(i, \lambda) \mid i \in I\} \cup \{(\lambda, j) \mid j \in J\}.$$

In other words, every set M' corresponds to at least one sequence of ways of transforming T_1 into T_2. Let

$$\Gamma_{(T_1,T_2)} = \{M' \mid M' \text{ is an extended mapping between } T_1 \text{ and } T_2\}.$$

Note that each mapping represents at least one sequence of edit operations which will transform T_1 and T_2. $\Gamma_{(T_1,T_2)}$ is merely an exhaustive enumeration of all these sequences. Note that in every element of $\Gamma_{(T_1,T_2)}$ the sibling and the ancestor orderings in both T_1 and T_2 are maintained. An example of $\Gamma_{(T_1,T_2)}$ is shown in Example II (See Figure VII). For example, the pair M_8 in Example II represents the edit operations of replacing "a" by "b" and "d" by "c", and inserting "e".

*** * * * * Insert Figure VII Here * * * * ***

3.4 The Abstract Measure $\Omega(T_1,T_2)$

The abstract measure $\Omega(T_1,T_2)$ between T_1 and T_2 induced by the system $(\mathcal{N},\tau,\omega)$, where $\tau = (T, \oplus, \otimes, \theta, I)$, is a map whose domain is $\mathcal{N}^* \times \mathcal{N}^*$ and whose range is T. Formally, we define

$$\Omega(T_1, T_2) = \begin{cases} I \text{ - the identity for } \otimes & \text{if } T_1 = T_2 = \mu; \\ \bigoplus_{M' \in \Gamma_{(T_1,T_2)}} \left[\bigotimes_{(x_i,y_i) \in M'} \omega(x_i,y_i) \right] & \text{otherwise.} \end{cases} \tag{3.1}$$

In general, $\Omega(T_1, T_2)$ need not be equal to $\Omega(T_2, T_1)$. The equality holds only when the function $\omega(.,.)$ inducing it obeys the following conditions :

$$\omega(a, b) = \omega(b, a) \qquad \text{for all } a, b \in \mathcal{N},$$
$$\omega(\lambda, a) = \omega(a, \lambda) \qquad \text{for all } a \in \mathcal{N}.$$

IV. NUMERIC MEASURES FOR $\Omega(T_1, T_2)$

We shall now show that various numeric measures involving T_1 and T_2 are obtained as special cases of $\Omega(T_1, T_2)$ by appropriately choosing concrete values for the abstract operators $\oplus$, $\otimes$ and for the function $\omega(.,.)$.

4.1 The distance between two trees

The distance between two trees T_1 and T_2, denoted by $\text{Dist}(T_1, T_2)$, is defined as the minimum of the sum of the elementary edit distances associated with the edit operations required to transform T_1 to T_2 ([Se77], [Ta79], [Lu79] and [ZS89]). The elementary edit distances themselves are specified in terms of a map $\omega_1(.,.)$ from $\tilde{\mathcal{N}} \times \tilde{\mathcal{N}}$ to R_p' obeying the following :

$$\omega_1(x, y) \geq 0; \quad \omega_1(x, x) = 0;$$
$$\omega_1(x, y) = \omega_1(y, x);$$
$$\omega_1(x, z) \leq \omega_1(x, y) + \omega_1(y, z).$$

The symmetry condition (requiring that $\omega_1(x,y) = \omega_1(y,x)$) need not necessarily be satisfied. But if it is, in general, the distance $Dist(T_1,T_2)$ is a strict distance metric, and a greater value of the indicates a greater dissimilarity between them.

THEOREM I : The quantity $Dist(T_1, T_2)$ between T_1 and T_2 is exactly the measure between them induced by the system $(\mathcal{N}, \tau_1, \omega_1)$, where τ_1 is the structure defined in Example III.

PROOF

The theorem is trivially true when $T_1 = T_2 = \mu$, since $Dist(\mu, \mu) = 0$. Consider the case when either T_1 or T_2 or both is not μ. From Section 3.3, we seen that the set of ways[1] by which T_1 can be edited to T_2 is given by the set $\Gamma_{(T_1,T_2)}$. Any extended map $M' \in \Gamma_{(T_1,T_2)}$ represents the edit operation of transforming x_i to y_i for all $(x_i, y_i) \in M'$. Hence the sum of the edit distances corresponding to this extended mapping, M', is given by the number $\sum_{(x_i,y_i)\in M'} \omega_1(x_i, y_i)$. By definition, the minimum of this quantity is the distance between T_1 and T_2. Since $\Gamma_{(T_1,T_2)}$ is the set of all extended mappings,

$$Dist(T_1, T_2) = \underset{M' \in \Gamma_{(T_1,T_2)}}{\mathrm{Min}} \left[\sum_{(x_i,y_i)\in M'} \omega_1(x_i, y_i) \right], \tag{4.1}$$

which is exactly the value of the measure between T_1 and T_2 induced by the system $(\mathcal{N}, \tau_1, \omega_1)$. Hence the theorem. ***

4.2 The Size of the Largest Common Sub-Tree

We shall now show that by using the operators $\oplus$ and $\otimes$ to represent the MAX(.,.) and the arithmetic addition operations, we can compute $SLCsT(T_1, T_2)$, the Size of the Largest Common Sub-Tree of T_1 and T_2 if the values of $\omega(.,.)$ are appropriately defined.

By definition, $SLCsT(T_1, T_2) = 0$ if $T_1 = T_2 = \mu$. Consider the nontrivial case when either T_1 or T_2 or both are not μ. Let Z_p' be the set consisting of the non-negative integers and ∞, and let $\omega_2(.,.)$ be a function defined from $\tilde{\mathcal{N}} \times \tilde{\mathcal{N}}$ to Z_p' by

$$\omega_2(a, b) = \begin{cases} 1 & (\text{if } a = b \neq \lambda,\ a,b \in \mathcal{N}); \\ 0 & \text{otherwise.} \end{cases}$$

Consider any extended mapping $M' \in \Gamma_{(T_1,T_2)}$. Corresponding to this extended mapping we can associate an integer W_2 where

[1]We refer the reader to [ZS89] for a formal proof which demonstrates the equivalence between edit sequences and mappings.

$$W_2 = \sum_{(x_i,y_i)\in M'} \omega_2(x_i, y_i).$$

By the definition of $\omega_2(.,.)$, the value of W_2 is exactly the number of non-λ pairs of nodes (x_i, y_i) in M' with $x_i = y_i$, which will yield a common Sub-Tree between T_1 and T_2. Note that this is true since the sibling and ancestor orders will be preserved by the extended mapping. Hence, W_2 is the size of a Common Sub-Tree. By the definition of $\Gamma_{(T_1,T_2)}$, the maximum value of W_2 is the value SLCsT(T_1, T_2). Hence

$$\text{SLCsT}(T_1, T_2) = \underset{M'\in \Gamma_{(T_1,T_2)}}{\text{Maximum}} \left[\sum_{(x_i,y_i)\in M'} \omega_2(x_i, y_i) \right]. \qquad (4.2)$$

This leads to the following theorem.

THEOREM II : Let τ_2 be the abstract structure (Z_p', MAX, +, 0, 0) where

i) Z_p' is the set consisting of the non-negative integers and ∞.

ii) MAX is the commutative binary operator defined by

$$\text{MAX}(p, q) = \begin{cases} p & \text{if } p \geq q; \\ q & \text{otherwise.} \end{cases}$$

iii) + represents arithmetic addition.

Then the SLCsT(T_1, T_2) is the measure induced by the system ($\mathcal{N}$, τ_2, ω_2).

PROOF :

The result follows from comparing (4.2) to the application of (3.1) to the ($\mathcal{N}$, τ_2, ω_2). ***

At this juncture, it is appropriate to distinguish between the Length of the Longest Common Subsequence Problem and the Size of the Largest Common Sub-Tree Problem. Due to the necessarity of preserving the Sibling and the Ancestor Properties of a tree, we can NOT compute the SLCsT(T_1, T_2) between two trees by merely computing the LLCS using merely the string representations of the corresponding trees. As an example, consider the trees shown in the Figure VIII. Since the postorder sequences of T_1 and T_2 are : "debca" and "bceda", respectively, an LLCS is "bca". However, no Common Sub-Tree of T_1 and T_2 may consist of both the nodes "b" and "c" simultaneously. In other words, trees obtained using the string representation of their LCSs **may** represent trees which cannot be their common Sub-Trees. More precisely, the sibling and the ancestor constraints of the "tree" will be totally ignored if we try to compute the SLCsT using any LCS algorithm. This is the reason why the problem currently studied is more complex.

* * * * *	**Insert Figure VIII Here**	* * * * *

4.3 Probability of erroneous trees

Consider the problem of transmitting a tree across a noisy channel. A pertinent question is that of evaluating the probability of receiving a tree T_2 given that a tree T_1 is transmitted[2]. In this section, we will show that this quantity can also be obtained as a special case of $\Omega(T_1, T_2)$ by appropriately choosing the concrete operators $\otimes$ and $\oplus$, and the specific function $\omega(.,.)$.

As a first attempt of solving the problem we consider the model analogous to the one used in [KO83] which causes only independent substitution and deletion errors. Let the elementary error probabilities (also called the "confusion" probabilities) be given by the function mapping $\tilde{N} \times \tilde{N}$ to [0, 1] as below:

(i) $p(b|a)$ is the conditional probability of receiving the node symbol "b" given that the node symbol "a" is transmitted.

(ii) $p(\lambda|a)$ is the conditional probability of the channel deleting the transmitted node symbol "a".

We define $p(a|\lambda) = 0$ for $a \in N$. The quantity $p(\lambda|\lambda)$ is undefined since it is not needed.

Let M' be any arbitrary element of $\Gamma_{(T_1,T_2)}$ and let

$$W_3 \quad = \quad \prod_{(x_i,y_i)\in M'} p(y_i|x_i), \tag{4.3}$$

where Π represents ***arithmetic*** multiplication.

W_3 is the product of the probabilities associated with the extended mapping M' given that every y_i was caused by the corresponding x_i. Since $p(a|\lambda) = 0$ for all $a \in N$, W_3 is equal to zero whenever the extended mapping M' represents the insertion of at least one node symbol in T_2.

We define $Prob(T_2|T_1)$ as the ***total*** probability of receiving T_2 given that T_1 is transmitted. $Prob(T_2|T_1)$ is defined as unity if both T_1 and T_2 are μ. Otherwise, it is defined as the sum of W_3 over all extended mappings M' in $\Gamma_{(T_1,T_2)}$. That is,

$$Prob(T_2|T_1) \quad = \quad \sum_{M'\in\Gamma_{(T_1,T_2)}} \left[\prod_{(x_i,y_i)\in M'} p(y_i|x_i)\right]. \tag{4.4}$$

The quantity $Prob(T_2|T_1)$ has the following properties.

THEOREM III : Let τ_3 be the abstract structure ([0, 1], +, *, 0, 1) where

i) [0, 1] is the set consisting of the non-negative real numbers between 0 and 1.

ii) + and * represent arithmetic addition and multiplication respectively.

[2]Here we assume that the tree itself is transmitted as a two dimensional entity. The actual serialization of this transmission process is not considered here.

Then, $\mathrm{Prob}(T_2|T_1)$, the ***probability of receiving T_2 when T_1 is transmitted***, is exactly the measure between T_1 and T_2 induced by the system $(\mathcal{N}, \tau_3, \omega_3)$, where $\omega_3(.,.)$ is defined by :

$$\omega_3(a, b) = p(b|a) \qquad \text{for all } a, b \in \tilde{\mathcal{N}}.$$

PROOF

The form of the expression obtained for $\mathrm{Prob}(T_2|T_1)$ (See (4.4)) can be seen to be the one which follows when (3.1) is applied to the system $(\mathcal{N}, \tau_3, \omega_3)$. It remains to be proved that the quantity defined by (4.4) is not just computationally well defined, but also a well defined ***probability*** measure. We shall now show that $\mathrm{Prob}(T_2|T_1)$ is a valid probability assignment for the probability of receiving T_2 given T_1 is transmitted, given that the channel only substitutes and deletes the node values.

Let $|T_1| = m$. Since $\mathrm{Prob}(T_2|T_1) \geq 0$, all that has to be proved is that

$$\sum_{T_2 \in \tilde{\mathcal{N}}_m} \mathrm{Prob}(T_2|T_1) = 1,$$

where $\tilde{\mathcal{N}}_m$ is the set of trees over $\mathcal{N}$ of size less than or equal to m.

Since the channel cannot cause insertion errors, $M' \in \Gamma_{(T_1,T_2)}$ will yield a non-zero contribution to $\mathrm{Prob}(T_2|T_1)$ if and only if $|M'| = m$. Let the subset of elements of $\Gamma_{(T_1,T_2)}$ which have $|M'| = m$ be Γ_m. Then,

$$\mathrm{Prob}(T_2|T_1) = \sum_{M' \in \Gamma_m} \left[\prod_{(x_i, y_i) \in M'} p(y_i|x_i) \right].$$

The sum on the R.H.S involves ***independently*** considering all the possible transformations of the individual symbols of T_1. Thus, the addition and multiplication symbols can be interchanged yielding

$$\sum_{T_2 \in \tilde{\mathcal{N}}_m} \mathrm{Prob}(T_2|T_1) = \prod_{i=1}^{m} \left[\sum_{v \in \tilde{\mathcal{N}}} p(v|x_i) \right].$$

which equals unity since $\sum_{v \in \tilde{\mathcal{N}}} p(v|x_i) = 1$ for every x_i, and the results follows. ***

4.4 Likelihood of erroneous trees

In the last section we considered the problem of transmitting a tree across a noisy channel which causes **no** insertion error. However, the problem of evaluating the total probability of receiving a tree T_2 given that a tree T_1 is transmitted across a noisy channel which causes *all* the three kinds of edit errors (including an arbitrary number of insertions) is not so direct, but on the contrary far from trivial (see [KO84] for the corresponding problem in the case of strings). This is because of the fact that if we do not restrict the number of possible insertion errors which can occur, the computation of the quantity $\sum_{T_2 \in \mathcal{N}^*} \text{Prob}(T_2|T_1)$, which is encountered in the process of justifying the model, will involve an infinite sum which is almost certainly divergent. In order to solve the problem in a more reasonable way, we shall consider the ***a posteriori*** approach in which we intend to evaluate the probability of a ***received*** tree T_2| having been caused by the transmission of a tree T_1, where T_1 itself is a memeber of a finite dictionary of trees.

Consider a channel which causes independent substitution, deletion, and insertion errors. Let the elementary error probabilities be given by the function mapping $\tilde{\mathcal{N}} \times \tilde{\mathcal{N}}$ to [0, 1] as below:

(i) $q(a \to b)$ is the conditional probability of receiving the node symbol "b" given that the node symbol "a" is transmitted.

(ii) $q(a \to \lambda)$ is the conditional probability of the channel deleting the transmitted node symbol "a".

(ii) $q(\lambda \to a)$ is the conditional probability of the channel inserting the symbol "a" during the transmission given that an insertion is taking place.

Since a transmitted symbol can either be transmitted correctly, deleted or transformed into another symbol, and since the inserted symbol can be any one of the symbol in $\mathcal{N}$, we have :

$$\sum_{v \in \tilde{\mathcal{N}}} q(a \to v) = 1 \qquad \text{for all } a \in \mathcal{N},$$

and

$$\sum_{v \in \mathcal{N}} q(\lambda \to v) = 1.$$

Observe again that, the quantity $p(\lambda|\lambda)$ is undefined since it is not needed.

Let M' be any arbitrary element of $\Gamma_{(T_1,T_2)}$ and let

$$W_4 = \prod_{(x_i,y_i) \in M'} q(x_i \to y_i),$$

where Π represents ***arithmetic*** multiplication. W_4 is the product of the probabilities associated with the extended mapping M' given that every y_i was caused by the corresponding x_i.

Let us define $\mathcal{L}(T_1 \to T_2)$ as the likelihood[3] of transmitting T_1 given that T_2 is received. $\mathcal{L}(T_1 \to T_2)$ is defined as unity if both T_1 and T_2 are μ. Otherwise, it is defined as the sum of W_4 over all extended mappings M' in $\Gamma_{(T_1,T_2)}$. That is,

[3]The quantity $\mathcal{L}(T_1|T_2)$ is termed as a ***likelihood*** because it is easy to see that it is not a consistent probability measure. However, viewed from an ***a posteriori*** perspective the normalized quantities can be viewed as probabilities.

$$\mathcal{L}(T_1 \to T_2) \quad = \quad \sum_{M' \in \Gamma_{(T_1, T_2)}} \left[\prod_{(x_i, y_i) \in M'} q(x_i \to y_i) \right]. \qquad (4.5)$$

The quantity $\mathcal{L}(T_1 \to T_2)$ has the following properties.

THEOREM IV : Let τ_3 be the abstract structure as defined in Theorem IV. Then, the quantity $\mathcal{L}(T_1 \to T_2)$, the likelihood of having T_1 being transmitted given T_2 is received, is exactly the measure between T_1 and T_2 induced by the system $(\mathcal{N}, \tau_3, \omega_4)$, where $\omega_4(.,.)$ is defined by :

$$\omega_4(a, b) = q(a \to b) \qquad \text{for all } a, b \in \tilde{\mathcal{N}}.$$

PROOF

The theorem follows directly by applying (3.1) to $(\mathcal{N}, \tau_3, \omega_4)$ and comparing the result with (4.5). ***

Note that the ***a priori*** likelihoods $\mathcal{L}(T_1 \to Q)$ do not constitute a valid probability assignment for the probability of ***receiving*** Q given that T_1 is transmitted. This is because we can express $\mathcal{L}(T_i \to Q)$ as the sum of $\mathcal{L}_j(T_i \to Q)$ for $(0 \le j \le |Q|)$, where $\mathcal{L}_j(T_i \to Q)$ is the likelihood of having transmitted T_i given that Q is received and that j insertion errors have taken place. Hence,

$$\sum_{Q \in \mathcal{N}^*} \mathcal{L}(T_1 \to Q) = \sum_{Q \in \mathcal{N}^*} \left[\sum_{j=0}^{|Q|} \mathcal{L}_j(T_1 \to Q) \right]$$

and since we are only dealing with positive quantities which are bounded by unity, the summations can be interchanged to yield :

$$\sum_{Q \in \mathcal{N}^*} \mathcal{L}(T_1 \to Q) = \sum_{j=0}^{\infty} \left[\sum_{Q \in Q^j} \mathcal{L}_j(T_1 \to Q) \right]$$

where Q^j is the set of trees which can be received if exactly j insertions have taken place. Thus,

$$\sum_{Q \in \mathcal{N}^*} \mathcal{L}(T_1 \to Q) = \sum_{j=1}^{\infty} \left[\sum_{Q \in Q^j} \mathcal{L}_j(T_1 \to Q) \right] + \sum_{Q \in Q^0} \mathcal{L}_0(T_1 \to Q)$$

However, from the last section, we see that $\sum_{Q \in Q^0} \mathcal{L}_0(T_1 \to Q) = 1$. Hence, we have

$$\sum_{Q \in \mathcal{N}^*} \mathcal{L}(T_1 \to Q) = \sum_{j=1}^{\infty} \left[\sum_{Q \in Q^j} \mathcal{L}_j(T_1 \to Q) \right] + 1 \geq 1.$$

However, by viewing the problem from an ***a posteriori*** perspective a valid probability measure can be well defined. To achieve this, let T_1 be known to be an element of a **finite** dictionary of trees, $\mathcal{H}$. Then, if Q is any received tree, we define :

$$P(T_i|Q) = \frac{\mathcal{L}(T_i \to Q)}{\sum_{T_j \in \mathcal{H}} \mathcal{L}(T_j \to Q)}, \qquad \text{for all } T_i \in \mathcal{H}.$$

We shall now show that the quantity $P(T_i|Q)$ is a valid probability assignment for the *a posteriori* probability of having transmitted $T_1 \in \mathcal{H}$ given that Q is received.

THEOREM V

The quantity $P(T_i|Q)$ is a valid probability assignment for the ***a posteriori*** probability of having transmitted $T_1 \in \mathcal{H}$ given that Q is received.

PROOF

Assume $|\mathcal{H}| = k$. Since $P(T_1|Q) \geq 0$, we essentially have to prove that

$$\sum_{T_i \in \mathcal{H}} P(T_i|Q) = 1.$$

But,

$$\begin{aligned}
\sum_{T_i \in \mathcal{H}} P(T_i|Q) &= \sum_{i=1}^{k} P(T_i|Q) \\
&= \sum_{i=1}^{k} \frac{\mathcal{L}(T_i \to Q)}{\sum_{j=1}^{k} \mathcal{L}(T_j \to Q)} \\
&= \frac{\sum_{i=1}^{k} \mathcal{L}(T_i \to Q)}{\sum_{j=1}^{k} \mathcal{L}(T_j \to Q)}
\end{aligned}$$

which equals unity since the identical summations in the numerator and denominator are finite. ***

V. COMPUTATION OF $\Omega(T_1, T_2)$.

5.1 Recursive Properties of $\Omega(T_1, T_2)$

Since we are working with the postorder numbering of the trees, $T_1[i'..i]$ and $T_2[j'..j]$ will generally (and almost often) be forests. Fortunately, the definitions of mapping and extended mapping for ordered forests are valid just as in the case of trees. For convenience, we use $\Gamma_{(F_1,F_2)}$ and $\Omega(F_1, F_2)$ to represent the set of all extended mappings and the abstract measure between the forests $F_1=T_1[i'..i]$ and $F_2=T_2[j'..j]$, respectively. Explicitly, we also use F_Meas($T_1[i'..i]$, $T_2[j'..j]$) to denote the abstract measure between the forest $T_1[i'..i]$ and the forest $T_2[j'..j]$. The abstract measure between the ***subtree*** of T_1 rooted at i and the ***subtree*** of T_2 rooted at j is denoted by T_Meas(i, j).

To derive the recursive properties of the abstract measure we shall now introduce an additional quantity, referred to as F_Meas_I(.,.). Given an extended mapping M', we say $T_2[j]$ belongs to M' if there is a k such that $(T_1[k],T_2[j])$ is in M, where M is the mapping over which M' is derived. We use F_Meas_I($T_1[i'..i]$,$T_2[j'..j]$) to denote the abstract measure over all extended mappings M' between $T_1[i'..i]$ and $T_2[j'..j]$ such that $T_2[j]$ belongs to M'. The reason why we have to introduce the latter measure is to take care of the scenario when $T_1[i]$ is not in the mapping but $T_2[j]$ is in the mapping. In that case, unlike in the straightforward tree-edit distance formualtion, it can be seen that :

$$\text{F\_Meas}(T_1[1..i], T_2[1..j]) \neq \text{F\_Meas}(T_1[1..i-1], T_2[1..j]) \otimes d(T_1[i],0).$$

The reason for this is that F_Meas($T_1[1..i-1]$, $T_2[1..j]$) will include the case where $T_2[j]$ is not in the mapping, although by assumption it is in the mapping.

Another quantity which will assist us in the compuatation is the quantity T'_Meas(i,j) which we shall now introduce. We define the quantity T_Meas(i,j) as :

$$\text{T'\_Meas}(i,j) = \text{F\_Meas}(T_1[\delta(i)..i-1], T_2[\delta(j)..j-1]).$$

Note that $T[\delta(i)..i-1]$ is the forest obtained by deleting the root of the subtree rooted at $T[i]$.

We now prove the analytic properties of the above quantities.

THEOREM VI : Let $i_1 \in$ Anc(i) and $j_1 \in$ Anc(j), then,

(i) $\text{F\_Meas}(\mu, \mu) = I$, the identity for $\otimes$.

(ii) $\text{F\_Meas}(T_1[\delta(i_1)..i], \mu) = \text{F\_Meas}(T_1[\delta(i_1)..i-1], \mu) \otimes \omega(T_1[i], \lambda)$.

(iii) $\text{F\_Meas}(\mu, T_2[\delta(j_1)..j]) = \text{F\_Meas}(\mu, T_2[\delta(j_1)..j-1]) \otimes \omega(\lambda, T_2[j])$.

PROOF

For case (i), since the set $\Gamma_{(F_1,F_2)}$ is empty, by the definition of $\Omega(F_1, F_2)$, F_Meas(μ, μ) $= I$, the identity for $\otimes$.

For case (ii), the set $\Gamma_{(F_1,F_2)}$ has only one element, namely $\{(T_1[\delta(i_1)],\lambda), (T_1[\delta(i_1)+1],\lambda), ..., (T_1[i],\lambda)\}$. Hence, by (3.1),

$$F\_Meas(T_1[\delta(i_1)..i], \mu) = \bigotimes_{j=\delta(i_1)}^{i} \omega(T_1[i],\lambda),$$

which by the associativity of $\otimes$ reduces to :

$$\bigotimes_{j=\delta(i_1)}^{i-1} \omega(T_1[j],\lambda) \otimes \omega(T_1[i], \lambda).$$

But $\bigotimes_{j=\delta(i_1)}^{i-1} \omega(T_1[j],\lambda) = F\_Meas(T_1[\delta(i_1)..i-1], \mu)$. Hence,

$$F\_Meas(T_1[\delta(i_1)..i], \mu) = F\_Meas(T_1[\delta(i_1)..i-1], \mu) \otimes \omega(T_1[i], \lambda),$$

and the result follows.

In an exactly similar way (iii) can also be proved. Hence the theorem. ***

The above equations form the boundary values of F_Meas(.,.) where either tree (forest) is empty. However, for the intermediate values of the F_Meas(.,.), the recursive formulation becomes much more complex, and cannot be merely obtained by a generalization of the corresponding results of the tree-editing problem derived in [ZS89]. This is because, unlike in the latter, in our case the measure F_Meas is intricately related to the quantity F_Meas_I defined earlier. The following theorems will demonstrate how the two quantites are related.

THEOREM VII : Let $i_1 \in Anc(i)$ and $j_1 \in Anc(j)$, then,

$$F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) =$$
$$F\_Meas\_I(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) \oplus F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j-1]) \otimes \omega(\lambda,T_2[j]).$$

PROOF

We compute $F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j])$ for $\delta(i_1) \le i \le i_1$ and $\delta(j_1) \le j \le j_1$. The mapping M between the forest $F_1=T_1[\delta(i_1)..i]$ and forest $F_2=T_2[\delta(j_1)..j]$ can be seen to include the node in the following two ways :

Case (1) : If $T_2[j]$ is in M.

Case (2) : If $T_2[j]$ is not in M.

Thus, if the individual terms evaluated for $F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j])$ for the above two cases are given for $1 \le k \le 2$ by $F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j])|_{Case(k)}$, then using the associativity of $\oplus$, we can see that :

$$F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = \bigoplus_{k=1}^{2} F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j])|_{Case(k)}.$$

We shall consider each of these cases separately.

Case (1) : It is clear that, by definition,

$$F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j])|_{Case(1)} = F\_Meas\_I(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]).$$

Case (2) : Let Γ_A be the maximal subset of $\Gamma_{(F_1,F_2)}$ such that if $M \in \Gamma_A$ then $(\lambda,T_2[j])$ belongs to M. By definition, we have,

$$F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j])|_{Case(2)} = \bigoplus_{M\in\Gamma_A} \left\{ \bigotimes_{(x_i,y_i)\in M} \omega(x_i,y_i) \right\}$$

Since $\otimes$ is commutative, the above expression reduces to :

$$\bigoplus_{M\in\Gamma_A} \left[\left\{ \bigotimes_{(x_i,y_i)\in M-\{(\lambda,T_2[j])\}} \omega(x_i,y_i) \right\} \otimes \omega(\lambda,T_2[j]) \right].$$

Because $\otimes$ distributes over $\oplus$, we have,

$$\left[\bigoplus_{M\in\Gamma_A} \left\{ \bigotimes_{(x_i,y_i)\in M-\{(\lambda,T_2[j])\}} \omega(x_i,y_i) \right\} \right] \otimes \omega(\lambda,T_2[j]).$$

This is equivalent to :

$$F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = F\_Meas(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j-1]) \otimes \omega(\lambda,T_2[j]),$$

and the theorem is proved. ***

From the above results we have shown how the original measure F_Meas(.,.) is recursively related to the measure which we "artificially" introduced, namely, F_Meas_I(.,.). We shall now show a "dual" result and prove that just like the original measure, the related quantity F_Meas_I(.,.) also has some interesting recursive properties -- it will be shown to be recursively related to the original index, F_Meas(.,.) itself.

THEOREM VIII : Let $i_1 \in Anc(i)$ and $j_1 \in Anc(j)$. Then, if $i=\delta(i_1)$,

$$F\_Meas\_I(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = F\_Meas(\mu,T_2[\delta(j_1)..j-1]) \otimes \omega(T_1[i],T_2[j]).$$

PROOF :

Since $i=\delta(i_1)$, $T_1[\delta(i_1)..i]=T_1[i]$. Since we know $T_2[j]$ belongs to every mapping, $(T_1[i],T_2[j])$ belongs to every mapping. Therefore,

$$F\_Meas\_I(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = F\_Meas(\mu,T_2[\delta(j_1)..j-1]) \otimes \omega(T_1[i],T_2[j]).$$ ***

THEOREM IX : Let $i_1 \in \mathrm{Anc}(i)$ and $j_1 \in \mathrm{Anc}(j)$. Then, if $i \neq \delta(i_1)$,

$$\begin{aligned}
&\mathrm{F\_Meas\_I}(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = \\
&\mathrm{F\_Meas\_I}(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j]) \otimes \omega(T_1[i],\lambda) \\
&\oplus \big[\mathrm{F\_Meas}(T_1[\delta(i_1)..\delta(i)-1],T_2[\delta(j_1)..\delta(j)-1]) \\
&\qquad \otimes \mathrm{F\_Meas}(T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]) \otimes \omega(T_1[i],T_2[j])\big].
\end{aligned}$$

PROOF :

Let $\Phi_{(F_1,F_2)}$ denote the set of all the extended mappings between $T_1[\delta(i_1)..i]$ and $T_2[\delta(j_1)..j]$ such that $T_2[j]$ belongs to each mapping. Consider now an arbitrary $M \in \Phi_{(F_1,F_2)}$. To evaluate the expression we have to consdier two mutually exclusive and exhaustive cases :

Case(1) : $T_1[i]$ is not in M

Case(2) : $T_1[i]$ is in M.

Let Γ_A be the maximal subset of $\Phi_{(F_1,F_2)}$ such that if $M \in \Gamma_A$ then $(T_1[i],\lambda) \in M$. Let,

$$\Gamma_B = \Phi_{(F_1,F_2)} - \Gamma_A.$$

Now,

$$\mathrm{F\_Meas\_I}(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = \underset{M \in \Phi_{(F_1,F_2)}}{\textcircled{\scriptsize Σ}} \left[\underset{(x,y) \in M}{\textcircled{\scriptsize Π}} \omega(x,y) \right]$$

$$= \left[\underset{M \in \Gamma_A}{\textcircled{\scriptsize Σ}} \left\{ \underset{(x,y) \in M}{\textcircled{\scriptsize Π}} \omega(x,y) \right\} \right] \oplus \left[\underset{M \in \Gamma_B}{\textcircled{\scriptsize Σ}} \left\{ \underset{(x,y) \in M}{\textcircled{\scriptsize Π}} \omega(x,y) \right\} \right].$$

We now consider these two cases separately.

Case(1) :

$$\left[\underset{M \in \Gamma_A}{\textcircled{\scriptsize Σ}} \left\{ \underset{(x,y) \in M}{\textcircled{\scriptsize Π}} \omega(x,y) \right\} \right]$$

$$= \underset{M \in \Gamma_A}{\textcircled{\scriptsize Σ}} \left[\left\{ \underset{(x,y) \in M - \{(T_1[i],\lambda)\}}{\textcircled{\scriptsize Π}} \omega(x,y) \right\} \otimes \omega(T_1[i],\lambda) \right]$$

$$= \left[\underset{M \in \Gamma_A}{\textcircled{\scriptsize Σ}} \left\{ \underset{(x,y) \in M - \{(T_1[i],\lambda)\}}{\textcircled{\scriptsize Π}} \omega(x,y) \right\} \right] \otimes \omega(T_1[i],\lambda),$$

where the latter two steps follow from the associativity of $\otimes$ and the distributivity of $\otimes$ over $\oplus$. Indeed, the latter, by definition is nothing but :

$$\mathrm{F\_Meas\_I}(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j]) \otimes \omega(T_1[i],\lambda).$$

Case(2) : Arguing as before,

$$\left[\bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in M} \omega(x,y) \right\} \right]$$

$$= \bigodot_{M \in \Gamma_B} \left[\left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in M - \{(T_1[i], T_2[j])\}} \omega(x,y) \right\} \otimes \omega(T_1[i], T_2[j]) \right]$$

$$= \left[\bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in M - \{(T_1[i], T_2[j])\}} \omega(x,y) \right\} \right] \otimes \omega(T_1[i], T_2[j]).$$

We now invoke the ancestor property of M to further partition each individual forest into a subforest and a tree respectively. We shall accomplish this by defining Z to be the maximal subset of $M-\{(T_1[i],T_2[j])\}$ such that $x \geq \delta(i)$ and $y \geq \delta(j)$ for all $(x,y) \in Z$. Note that, due to the ancestor property of M, all the remaining elements in the set $M-\{(T_1[i],T_2[j])\}-Z$ satisfy the condition :

$x \leq \delta(i)$ and $y \leq \delta(j)$.

Using the associativity of $\otimes$ we further reduce the above to :

$$\left[\bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in M - \{(T_1[i], T_2[j])\} - Z} \omega(x,y) \right\} \otimes \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in Z} \omega(x,y) \right\} \right] \otimes \omega(T_1[i], T_2[j])$$

$$= \left[\bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in M - \{(T_1[i], T_2[j])\} - Z} \omega(x,y) \right\} \otimes \bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in Z} \omega(x,y) \right\} \right]$$

$$\otimes\ \omega(T_1[i], T_2[j])$$

$$= \left[\bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in M - \{(T_1[i], T_2[j])\} - Z} \omega(x,y) \right\} \right] \otimes \left[\bigodot_{M \in \Gamma_B} \left\{ \bigcirc\!\!\!\!\prod_{(x,y) \in Z} \omega(x,y) \right\} \right]$$

$$\otimes\ \omega(T_1[i], T_2[j]).$$

Consider now the sets $\Gamma_{(F_1', F_2')}$ and $\Gamma_{(F_1'', F_2'')}$, where,

$F_1' =$	$T_1[\delta(i_1)..\delta(i)-1]$,	$F_2' =$	$T_2[\delta(j_1)..\delta(j)]$,
$F_1'' =$	$T_1[\delta(i)..i-1]$, and,	$F_2'' =$	$T_2[\delta(j)..j-1]$.

By comparing the elements between the sets Γ_B and $\Gamma_{(F_1',F_2')}$ and between Γ_B and $\Gamma_{(F_1'',F_2'')}$, we observe that: :

$$\left[\bigcircledSigma_{M\in\Gamma_B}\left\{\bigcircledPi_{(x,y)\in M-\{(T_1[i],T_2[j])\}-Z}\omega(x,y)\right\}\right]=\left[\bigcircledSigma_{M\in\Gamma_{(F_1',F_2')}}\left\{\bigcircledPi_{(x,y)\in M}\omega(x,y)\right\}\right],$$

and also that :

$$\left[\bigcircledSigma_{M\in\Gamma_B}\left\{\bigcircledPi_{(x,y)\in Z}\omega(x,y)\right\}\right]=\left[\bigcircledSigma_{M\in\Gamma_{(F_1'',F_2'')}}\left\{\bigcircledPi_{(x,y)\in M}\omega(x,y)\right\}\right].$$

This implies that

$$\left[\bigcircledSigma_{M\in\Gamma_B}\left\{\bigcircledPi_{(x,y)\in M}\omega(x,y)\right\}\right] = \text{F\_Meas}(T_1[\delta(i_1)..\delta(i)-1],T_2[\delta(j_1)..\delta(j)-1])$$

$$\otimes\ \text{F\_Meas}(T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]) \otimes \omega(T_1[i],T_2[j]).$$

Hence the theorem. ***

It is easy to construct a recursive algorithm by using the above results. However, both the time and space complexities of the algorithm will be prohibitively large. Improvements in both the time and space complexities can be obtained by utilizing the following result which considers the case when the subforests are themselves entire subtrees.

THEOREM X : Let $i_1 \in \text{Anc}(i)$, $j_1 \in \text{Anc}(j)$ and $i \neq \delta(i_1)$. Then, under the following conditions, F_Meas_I can be computed using the following formulae :

If $\delta(i)=\delta(i_1)$ and $\delta(j)=\delta(j_1)$, then :

$$\text{F\_Meas\_I}(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = \text{F\_Meas\_I}(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j]) \otimes \omega(T_1[i],\lambda)$$
$$\oplus\ \text{F\_Meas}(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j-1]) \otimes \omega(T_1[i],T_2[j]).$$

Else (**i.e.,** If $\delta(i) \neq \delta(i_1)$ or $\delta(j) \neq \delta(j_1)$)

$$\text{F\_Meas\_I}(T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]) = \text{F\_Meas\_I}(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j]) \otimes \omega(T_1[i],\lambda)$$
$$\oplus \Big[\text{F\_Meas}(T_1[\delta(i_1)..\delta(i)-1],T_2[\delta(j_1)..\delta(j)-1])$$
$$\otimes\ \text{F\_Meas}(T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]) \otimes \omega(T_1[i],T_2[j])\Big].$$

PROOF

We only need to prove the first "If" part .

Note that if $\delta(i)=\delta(i_1)$ and $\delta(j)=\delta(j_1)$, then,

$F\_Meas(T_1[\delta(i_1)..\delta(i)-1],T_2[\delta(j_1)..\delta(j)-1]) = F\_Meas(\mu,\mu) = I$, where I is the identity of $\otimes$.
Also, in this situation,

$$F\_Meas(T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]) = F\_Meas(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j-1]).$$

Therefore,

$$\begin{aligned} &F\_Meas(T_1[\delta(i_1)..\delta(i)-1],T_2[\delta(j_1)..\delta(j)-1]) \otimes F\_Meas(T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]) \otimes \omega(T_1[i],T_2[j]) \\ &\quad = I \otimes F\_Meas(T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]) \otimes \omega(T_1[i],T_2[j]) \\ &\quad = F\_Meas(T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j-1]) \otimes \omega(T_1[i],T_2[j]), \end{aligned}$$

and the theorem is proved. ***

5.2 A Dynamic Programming Algorithm to Compute $\Omega(T_1, T_2)$

Theorems VII to X suggest that we can use a dynamic programming flavored algorithm to solve the general tree comparison problem. First of all, note that the second part of Theorem X suggests that if we are to compute the quantity $T\_Meas(i_1,j_1)$, we have to have available the quantity $T'\_Meas(i,j)$, which is obtained when we compute $T\_Meas(i,j)$, for all i and j such that the node i and j are descendants of i_1 and j_1 except nodes on the path from i_1 to $\delta(i_1)$ and the nodes on the path from j_1 to $\delta(j_1)$. Furthermore the first part of Theorem X asserts that the measures associated with the nodes on the path from i_1 to $\delta(i_1)$ and from j_1 to $\delta(j_1)$ get computed (as a by-product) in the process of computing the $T\_Meas(i_1,j_1)$. Indeed, the set of nodes for which the computation of T_Meas(.,.) must be done independently before the T_Meas(.,.) associated with their ancestors can be computed is called the set of "Essential_Nodes". This suggests a bottom-up approach for computing the T_Meas(.,.) between all pairs of subtrees.

To formally present the algorithm, we define the set of Essential_Nodes described above.

Definition : For any tree T, the set Essential_Nodes[4] is defined as :

$$Essential\_Nodes(T) = \{k \mid \text{there exists no } k' > k \text{ such that } \delta(k) = \delta(k')\}.$$

Informally speaking, if k is in Essential_Nodes(T) then either k is the root or k has a left sibling. Intuitively, this set will be the roots of all subtrees of tree T that need separate computations. As an example, for the trees shown in Figure VI, the $Essential\_Nodes(T_1) = \{2,6,7,8\}$ and the $Essential\_Nodes(T_2) = \{4,5,6\}$.

It is easy to see that the function $\delta()$ and the set Essential_Nodes() can be computed in linear time. We assume that the results are stored in arrays $\delta[]$ and Essential_Nodes[] respectively.

[4]The set of nodes which we refer to as the set of Essential_Nodes happens to be exactly the same as the set of nodes defined in [ZS89] as the LR_keyroots set. Although these sets are identical, the implication of a node being in the sets is slightly different because in our case the evaluation of F_Meas involves quantities which are defined in terms of F_Meas_I. However, in both our result and the results of [ZS89], the computation needs to be explicitly done only for these nodes.

Furthermore, we assume that the elements in array Essential_Nodes [] are sorted and stored in an increasing order as per the postorder representation.

The main algorithm Abstract_T_Measure, essentially computes the measure between the overall trees by invoking a procedure Compute_Abstract_T_Meas between every pair of nodes in the Essential_Nodes of the trees. The latter procedure is devised by using dynamic programming. To achieve this, we shall use two permanent arrays T_Meas[i,j] and T'_Meas[i,j] for F_Meas($T_1[\delta(i)..i],T_2[\delta(j)..j]$) and F_Meas($T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]$) respectively. We will also use two other temporary arrays to store the quantities F_Meas($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$) and F_Meas_I($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$) respectively. Note that after the array T_Meas[.,.] has been computed, the value of the abstract measure $\Omega(T_1,T_2)$ between the trees T_1 and T_2 is stored in the memory location T_Meas[$|T_1|,|T_2|$].

The formal algorithm follows.

ALGORITHM Abstract_T_Measure

INPUT : Trees T_1 and T_2 and the set of elementary edit measures.

OUTPUT : $\Omega(T_1,T_2)$

ASSUMPTION :

The algorithm assumes a procedure **Preprocess (T_1, T_2)** which preprocesses the trees and yields the δ[] and Essential_Nodes[] arrays for both trees. These quantities are assumed to be global.
Note that the algorithm invokes **Compute_Abstract_T_Meas()**.

```
BEGIN {Abstract_T_Meas }
        For i' ← 1 to |Essential_Nodes_1| Do
            For j' ← 1 to |Essential_Nodes_2| Do
                i ← Essential_Nodes_1[i']
                j ← Essential_Nodes_2[j']
                Compute_Abstract_T_Meas(i,j)
            EndFor
        EndFor
        $\Omega(T_1, T_2) \leftarrow$ T_Meas[$|T_1|,|T_2|$]
END Algorithm Abstract_T_Measure

PROCEDURE   Compute_Abstract_T_Meas($i_1,j_1$)
BEGIN
        F_Meas($\mu,\mu$) ← $I$, the identity of $\otimes$
        For i ← $\delta(i_1)$ .. $i_1$ Do
            F_Meas($T_1[\delta(i_1)..i], \mu$) ← F_Meas($T_1[\delta(i_1)..i-1], \mu$) $\otimes$ $\omega(T_1[i],\lambda)$
        For j ← $\delta(j_1)$ .. $j_1$ Do
            F_Meas($\mu, T_2[\delta(j_1)..j]$) ← F_Meas($\mu, T_2[\delta(j_1)..j-1]$) $\otimes$ $\omega(\lambda,T_2[j])$
        For i ← $\delta(i_1)$ .. $i_1$ Do
            For j ← $\delta(j_1)$ .. $j_1$ Do
                If i = $\delta(i_1)$ Then
                    F_Meas_I($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$) ←
                                F_Meas($\mu,T_2[\delta(j_1)..j-1]$) $\otimes$ $\omega(T_1[i],T_2[j])$
                Else If $\delta(i) = \delta(i_1)$ and $\delta(j) = \delta(j_1)$ Then
                        F_Meas_I($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$) ←
                              [ F_Meas_I($T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j]$) $\otimes$ $\omega(T_1[i],\lambda)$]
                          $\oplus$  [ F_Meas($T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j-1]$) $\otimes$ $\omega(T_1[i],T_2[j])$]
                Else
                        F_Meas_I($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$) ←
                              [ F_Meas_I($T_1[\delta(i_1)..i-1],T_2[\delta(j_1)..j]$) $\otimes$ $\omega(T_1[i],\lambda)$]
                          $\oplus$  [   F_Meas($T_1[\delta(i_1)..\delta(i)-1],T_2[\delta(j_1)..\delta(j)-1]$)
                                 $\otimes$ T'_Meas[i,j] $\otimes$ $\omega(T_1[i],T_2[j])$]
                EndIf
                F_Meas($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$) ←
                    F_Meas_I($T_1[\delta(i_1)..i],T_2[\delta(j_1)..j]$)$\oplus$F_Meas($T_1[\delta(i)..i],T_2[\delta(j)..j-1]$)$\otimes\omega(\lambda,T_2[j])$
                If $\delta(i) = \delta(i_1)$ and $\delta(j) = \delta(j_1)$ Then
                    T_Meas(i,j) ← F_Meas($T_1[\delta(i)..i],T_2[\delta(j)..j]$)
                    T'_Meas(i,j) ← F_Meas($T_1[\delta(i)..i-1],T_2[\delta(j)..j-1]$)
                EndIf
END   Procedure Compute_Abstract_T_Meas
```

5.3 Discussion of the Results

As mentioned in the introduction, the original novel concept of utilizing generalized operators to compute numeric and non-numeric quantities on a graph was first proposed by Aho, Hopcroft and Ullman in their excellent book [AHU74]. They did this by introducing the "closed semi-ring" model of computation. From a naive perspective it may appear that the results of this paper essentially utilize these concepts in the setting of the tree-editing problem. This is ***definitely*** not the case because, although the pioneering work in [AHU74] explains how the semi-ring can be defined, it does not consider the possibility that the topological structure of the underlying graph may prohibit the abstractions of the computations. Indeed, our results do not merely have pedagogical significance, since we show that the general abstraction of the tree-editing problem is not possible using the closed semi-ring model because tree transformations require the simultaneous maintenance of the parent and sibling properties of the respective trees.

In this sub-section we shall explain how the work in this paper differs from all the other related work that is currently reported.

In an earlier work, Kashyap and Oommen [KO83] demonstrated how a generalization of the closed semi-ring model could be utilized in the case of the string editing problem. They did this by defining, $\gamma_{A,B}$, the set of all ways by which a string A could be edited into another string, B. The strategy used in [KO83] was essentially one of growing the latter set as the lengths of the prefixes of the strings increased. The underlying graph obtained was therefore a trellis, and all the string edit operations could be preserved as the trellis was traversed.

A little insight will demonstrate that there is a significant difference between $\gamma_{A,B}$, the set of all edit operations defined for strings A and B in [KO82, KO83, KO84] and the corresponding set $\Gamma_{(T_1,T_2)}$ defined for trees T_1 and T_2. Consider the case when we are editing the prefixes of two strings A and B. Let these prefixes be A[1..i] to B[1..j]. Now consider the case when A[i] and B[j] are both not in the trace (i.e., A[i] is not being substituted for by B[j]). Then in the case of strings the corresponding set of edit operations will have two distinct cases in $\gamma_{A[1..i],B[1..j]}$:

(i) $\gamma_{A[1..i-1],B[1..j-1]}$ followed by $(A[i] \rightarrow \lambda)$ followed by $(\lambda \rightarrow B[j])$

(ii) $\gamma_{A[1..i-1],B[1..j-1]}$ followed by $(\lambda \rightarrow B[j])$ followed by $(A[i] \rightarrow \lambda)$

Indeed, whereas in Case(i) we first delete A[i] and then insert B[j], in Case (ii) we opt to first insert B[j] first and then delete A[i]. Note that both these cases are in $\gamma_{A[1..i],B[1..j]}$.

The whole picture changes in the case when we are editing trees (or rather forests). Consider the case when we are editing the forest $T_1[1..i]$ into $T_2[1..j]$. If $T_1[i]$ and $T_2[j]$ are both not in the mapping, then, in general, the only order for $(T_1[i] \rightarrow \lambda)$ and $(\lambda \rightarrow T_2[j])$ is :

First perform $(T_1[i] \rightarrow \lambda)$ and subsequently perform $(\lambda \rightarrow T_2[j])$.

This is because, in general, in the case of forests, it is not always possible to insert $T_2[j]$ prior to deleting T1[i]. Thus, the "growing of the set" is not possible since the topological properties of the set must be preserved. Consequently, the closed semi-ring properties of [AHU74] are not directly applicable.

This is also the reason why all the computations which involve comparing trees in which the insertion of a node is required, is not so straightforward. Indeed, the question of inserting a node is not so elementary as in the case of strings, because, in the latter case, inserting a symbol retains a prefix as a prefix. But in the case of trees, the insertion must be achieved to preserve the sibling and ancestor properties, and this is not achievable by every possible insertion. Here too, the closed semi-ring model would not be directly applicable.

We would also like to mention, that the closed semi-ring model of [AHU74] does not permit arithmetic addition and arithmetic multiplication as concrete examples of $\oplus$ and $\otimes$. Our model and the one used in [KO83] permit them. But unlike the model in [KO83] we have further required that $\otimes$ is commutative, since in the case of trees, the ***order*** in which the edit operations are performed is of significance.

It must finally be noted that our algorithm is not merely an abstract generalization of the earlier work of Zhang and Shasha [ZS89]. Unlike the traditional edit distance, the abstract measure defined in this paper is defined by repeatedly using $\oplus$ over all extended mappings, and thus, in the corresponding proofs, care must be taken to ensure that each such mapping is accounted for exactly once. Also, the abstract measure F_Meas does not possess the inherent recursive properties of the tree edit distance. This has forced us to introduce an additional quantity F_Meas_I. Indeed, as mentioned in Section IV, the reason why we have had to introduce the latter measure is to take care of the scenario when $T_1[i]$ is not in the mapping but $T_2[j]$ is in the mapping. This too is a consequence of generalizing the operators that are traditionally used in the tree editing problem. Note that in the case of the tree editing problem the corresponding quantity for F_Meas_I does not have to be introduced since the concrete occurrence of the $\oplus$ operator is merely the MIN operator.

VI. TIME AND SPACE COMPLEXITIES

We shall now analyze the time and space complexities of the algorithm. From [ZS89], we have the following result :

THEOREM XI

For every tree T, Cardinality (Essential_Nodes (T)) $\leq$ |Leaves(T)|.

PROOF

Due to the equivalence of the sets Essential_Nodes(T) and LR_keyroots(T) of [ZS89], the proof is identical to the proof of Lemma 6 of [ZS89] and is omitted here. ***

This leads us to our primary complexity result.

THEOREM XII

If for a tree T, Span(T) is defined as the Min{Depth(T), Leaves(T)}, the time and space complexities of this algorithm are :

Time : $O(|T_1| * |T_2| * \text{Span}(T_1) * \text{Span}(T_2))$ operations of $\oplus$ and $\otimes$.

Space : $O(|T_1| * |T_2|)$.

PROOF :

For space complexity, we use the arrays for F_Meas, T_Meas, T'_Meas and F_Meas_I. Clearly, each of these three arrays requires space $O(|T_1| * |T_2|)$.

With regard to the time complexity, we first observe that the preprocessing takes linear time to precompute the arrays δ[] and Essential_Nodes[] for both trees. Also, note that if the array T_Meas[.,.] is computed, the abstract measure, $\Omega(T_1, T_2)$, can be computed in time O(1) by looking at the content of the array directly. Hence, the dominant term involves computing the arrays T_Meas(i,j) and T_Meas_S(i,j) for all relevant i, and j. This dynamic programming algorithm involving the subtrees rooted at i and j, respectively, involves : O(Size(i) * Size(j)) computations. Therefore, total time required is :

$$\sum_{i=1}^{|\text{Essential\_Nodes}(T_1)|} \sum_{j=1}^{|\text{Essential\_Nodes}(T_2)|} \text{Size}(i) * \text{Size}(j)$$

$$\leq \sum_{i=1}^{|\text{Essential\_Nodes}(T_1)|} \text{Size}(i) * \sum_{j=1}^{|\text{Essential\_Nodes}(T_2)|} \text{Size}(j)$$

Since the sets Essential_Nodes is identical to the set defined in [ZS89] as the LR_keyroots, we see from Theorem 2 of [ZS89] that :

$$\sum_{i=1}^{|\text{Essential\_Nodes}(T)|} \text{Size}(i) \leq |T| * \text{Min}\{\text{Depth}(T), \text{Leaves}(T)\}$$

Since Span(T) is equal to Min{Depth(T), Leaves(T)}, and since every single computation here requires at most three invocations of $\otimes$ and two invocations of $\otimes$, the time complexity of the algorithm is $O(|T_1| * |T_2| * \text{Span}(T_1) * \text{Span}(T_2))$, and the theorem is proved ***

VII. SUMMARY, CONCLUSIONS AND OPEN PROBLEMS

In this paper we have defined an abstract measure of comparison, $\Omega(T_1, T_2)$, between two trees T_1 and T_2. This measure has been presented in terms of a set of elementary inter-symbol measures $\omega(.,.)$ and two abstract operators $\oplus$ and $\otimes$. By appropriately choosing the concrete values for these two operators and for $\omega(.,.)$, this measure can be used to define the edit distance between two trees and the Size of their Largest Common Sub-Tree (SLCsT). Additionally, a special case of this measure can also be used to quantify the probability $\text{Prob}(T_2|T_1)$, the probability of receiving T_2 given that T_1 was transmitted across a channel causing independent substitution and deletion errors. Finally, the same abstract measure can be used to quantify the probability of T_1 being the transmitted tree given that T_2 is the received tree containing independent substitution, insertion and deletion errors conditioned on the fact that the transmitted tree is an element from a finite dictionary, $\mathcal{H}$. Apart from the definition and formulation of the abstract measure $\Omega(T_1, T_2)$, we have also derived its recursive properties and an iterative dynamic programming scheme to compute it has been presented. The space and time complexities of the algorithm have been analyzed, where the latter is derived in terms of the time taken to execute the individual abstract operations.

The actual physical quantities represented by $\Omega(T_1, T_2)$ is summarized by Table I below for various concrete possibilities for the operators $\oplus$ and $\otimes$, and for the function $\omega(.,.)$.

***** **Insert Table I Here** *****

In conclusion, we would like to mention that we believe that techniques analogous to those introduced in this paper can be utilized to define and compute various ***nonnumeric*** (symbolic) measures between two trees. This is not so directly obtained as in the case of strings [KO83], because in the case of trees it appears as if we need two ***independent*** abstract operators for "$\otimes$" to handle the insertion operation. Indeed, this is to ensure that although the physical insertion operation maintains the ancestor and sibling properties of the trees consistently, the corresponding abstract operator "$\otimes$" used to define the abstract measure must be defined in such a way so as to simultaneously maintain these properties. This problem is currently being investigated. We are also currently investigating the use of these generalized measures in classification of noisy biological macromolecules.

REFERENCES

[AHU74] A. V. Aho, D. S. Hirschberg and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[AHU76] A. V. Aho, D. S. Hirschberg and J. D. Ullman, "Bounds on the complexity of the longest common subsequence problem", *J. ACM*, Vol 23 : pp 1-12 (Jan 1976).

[Al67] C. N. Albegra, "String similarity and misspellings", *Comm. ACM*, Vol 10 : pp 302-313 (May 1967).

[CL85] Y. C. Cheng and S. Y. Lu, "Waveform correlation by tree matching", *IEEE Trans. PAMI,* Vol : PAMI 7, pp 299-305 (1985).

[HD80] P. A. V. Hall and G. R. Dowling, "Approximate string matching", *Comput. Sur.*, Vol 12 : pp 381-402 (Dec 1980).

[Hi75] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences", *Comm. ACM*, Vol 18 : pp 341-343 (June 1975).

[Hi77] D. S. Hirschberg, "Algorithms for the longest common subsequence problem", *J. ACM*, Vol 24 : pp 664-675 (Oct 1977).

[Hi78] D. S. Hirschberg, "An information-theoretic lower bound for the longest common subsequence problem", *Inf. Proc. Letterrs*, Vol 7 : pp 40-41 (Jan 1978).

[HS77] J. W. Hunt and T. G. Szymanski, "A fast algorithm for computing longest common subsequences", *Comm. ACM*, Vol 20 : pp 350-353 (May 1977).

[KO82] R. L. Kashyap and B. J. Oommen, "Probabilistic correction of strings", *Proc. of the IEEE Trans. on Pattern Recognition and Image Processing*, pp 28-33 (June 1982).

[KO83] R. L. Kashyap and B. J. Oommen, "A common basic for similarity measures involving two strings", *Intern. J. Computer Math.*, Vol 13 : pp 17-40 (1983).

[KO84] R. L. Kashyap and B. J. Oommen, "Spelling correction using probabilistic methods", *Pattern Recognition Letters*, Vol 2 : pp 147-154 (1984).

[KT82] M. Kunze and G. Thierrin, "Maximal common subsequences of pairs of strings", *Congressus Numerantium*, Vol 34 : pp 299-311 (1982).

[LW75] R. Lowrance and R. A. Wagner, "An extension of the string-to-string correction problem", *J. ACM*, Vol 22: pp 177-183 (April 1975).

[Lu79] S. Y. Lu, "A tree-to-tree distance and its application to cluster analysis", *IEEE Trans. PAMI,* Vol : PAMI 1, No. 2, pp 219-224 (April 1979).

[Lu84] S. Y. Lu, "A tree-matching algorithm based on node splitting and merging", *IEEE Trans. PAMI,* Vol : PAMI 6, No 2, pp 249-256 (1984).

[Ma78] D. Maier, "The complexity of some problems on subsequences and superSequences", *J. ACM*, Vol 25 : pp 322-336 (April 1978).

[MP80] W. J. Maser and M. S. Paterson, "A faster algorithm computing string edit distances", *J. of Comp. and Syst. Sci.*, Vol 20 : pp 18-31 (1980).

[Se77] S. M. Selkow, "The tree-to-tree editing problem", *Inf. Proc. Letterrs*, Vol 6, No. 6, pp 184-186 (Dec. 1977).

[SK83] D. Sankoff and J. B. Kruskal, *Time Warps,String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, Addison-Wesley (1983).

[SZ90] B. Shapiro and K. Zhang, "Comparing multiple RNA secondary structures using tree comparisons", *Comp. Appl. Biosci.* Vol. 6, No. 4, pp. 309-318, (1990).

[Ta79] K. C. Tai, "The tree-to-tree correction problem", *J. ACM*, Vol 26, pp 422-433 (1979).

[TT88] E. Tanaka and K. Tanaka, " The tree-to-tree editing problem", *Intern. J. Pat. Recog.*, Vol 2, No. 2, : pp 221-240 (1988).

[WF74] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem", *J. ACM*, Vol 21 : pp 168-173 (Jan 1974).

[ZS89] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems", *SIAM J. Comput.*, Vol : 18, No. 6, pp 1245-1262 (Dec. 1989).

LIST OF FIGURES AND TABLES

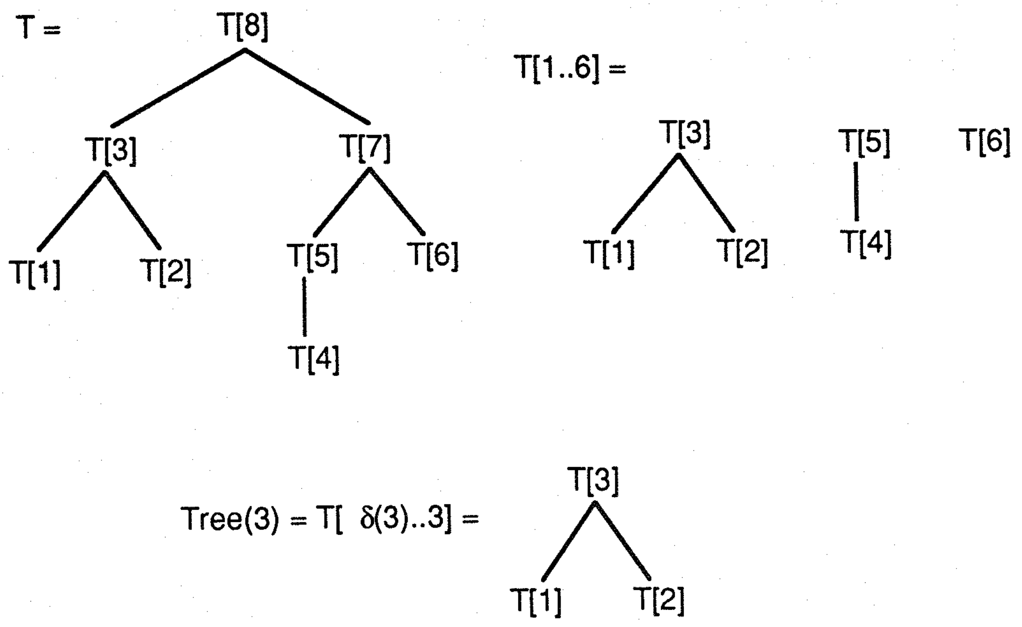


Figure I : An example of a tree, a postorder forest, a subtree, and the associated notations.

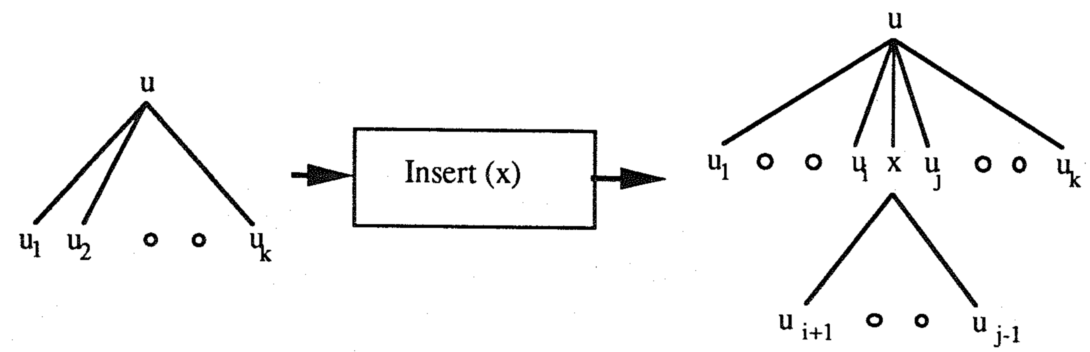


Figure II : An example of the insertion of a node.

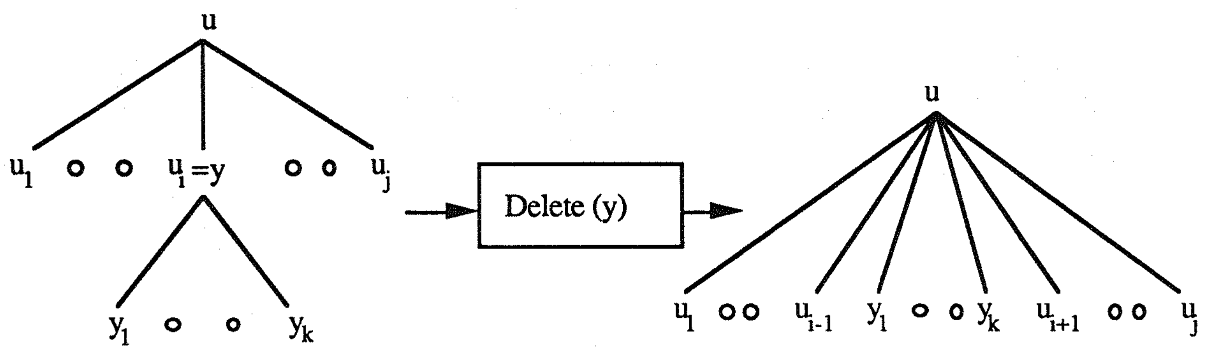

u
u_1 o o u_i=y o o u_j
y_1 o o y_k
Delete (y)
u
u_1 o o u_{i-1} y_1 o o y_k u_{i+1} o o u_j

Figure III : An example of the deletion of a node.

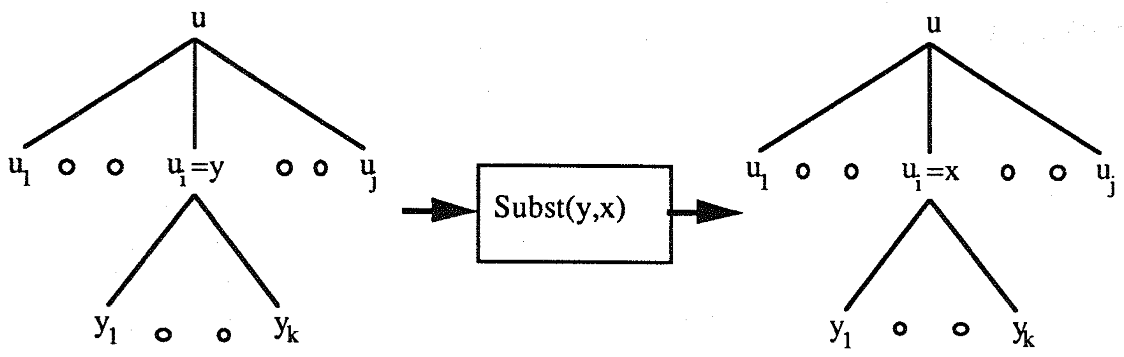
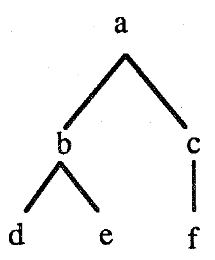


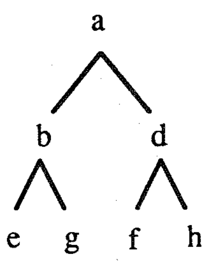
Figure IV : An example of the substitution of a node by another.

EXAMPLE I :

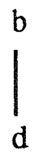
Let $T_2 =$



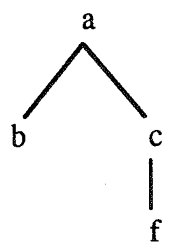
$T_3 =$



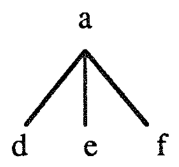
Then



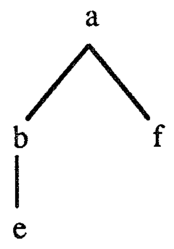
;



and



are some examples of the subsequence-tree of T_2 and



is a LCsT of T_2 and T_3 with SLCsT $(T_2,T_3) = 4$.

Figure V : Examples of Sub-Trees, LCsT and SLCsT.

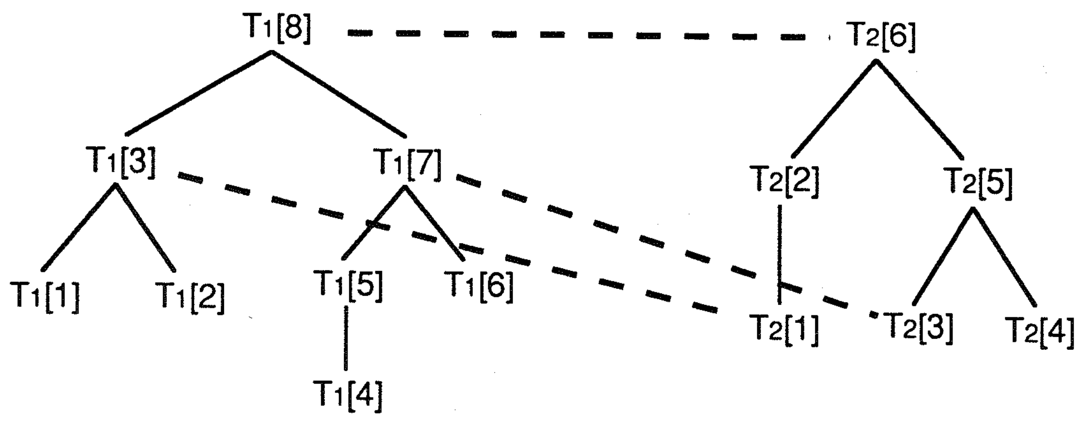
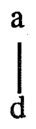


Figure VI : An example of a mapping.

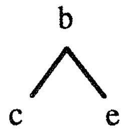
EXAMPLE II :

Let

$T_1 =$

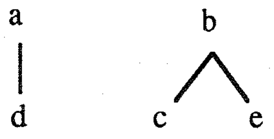


$T_2 =$

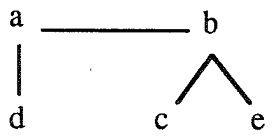


then, $\Gamma_{T1,T2} = \{M_i \mid 1 \leq i \leq 9\}$ where

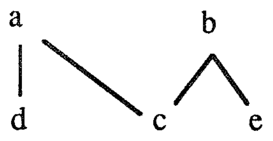
$M_1 =$



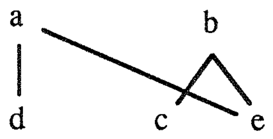
$M_2 =$



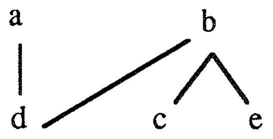
$M_3 =$



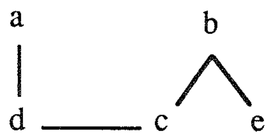
$M_4 =$



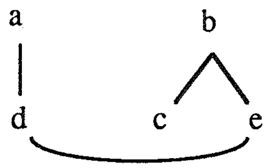
$M_5 =$



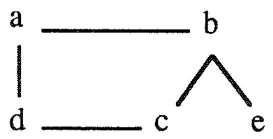
$M_6 =$



$M_7 =$



$M_8 =$



$M_9 =$

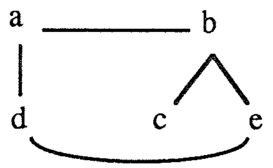


Figure VII : Example of an extended mapping

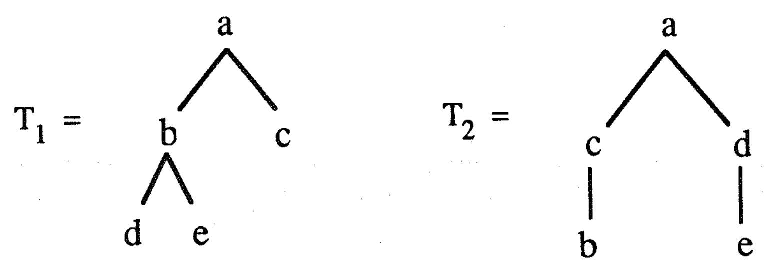


Figure VIII : Trees used to distinguish the difference between LLCS and SLCsT.

Problem	$\tau = (T, \oplus, \otimes, \theta, I)$	$\omega(.,.)$
Size of LCsT	$(Z_p', \text{MAX}, +, 0, 0)$	$\omega(a,b) = 1$, if $a=b\neq\lambda$, $\omega(a,b) = 0$, if $a\neq b\neq\lambda$, $\omega(a,\lambda) = 0$, and $\omega(\lambda,a) = 0$.
Probability of erroneous trees with substituion and deletion errors.	$([0, 1], +, *, 0, 1)$	$\omega(a,b) = p(b\|a)$.
Likelihood of erroneous trees with all three types of errors. From this the *a posteriori* probabilities can be defined.	$([0, 1], +, *, 0, 1)$	$\omega(a,b) = q(a\to b)$.
Distance[1]	$(Z_p', \text{MIN}, +, 0, 0)$	$\omega(a,b) = 0$, if $a=b\neq\lambda$, $\omega(a,b) = 1$, if $a\neq b\neq\lambda$, $\omega(a,\lambda) = 1$, and $\omega(\lambda,a) = 1$.

Table I : Summary of the use of generalized operators and generalized edit measures to evaluate various Tree-to-Tree similarity measures.

[1]In this case our algorithm is slightly more involved than the algorithm in [ZS89] due to the introduction of the additional quantity F_Meas_I. But in this case the latter quantity does not have to be introduced since the concrete occurrence of the $\oplus$ operator is the MIN operator.

School of Computer Science, Carleton University
Recent Technical Reports

TR-184	**Generating Binary Trees at Random** M.D. Atkinson and J.-R. Sack, December 1990
TR-185	**Uniform Generation of Combinatorial Objects in Parallel** M.D. Atkinson and J.-R. Sack, January 1991
TR-186	**Reduced Constants for Simple Cycle Graph Separation** Hristo N. Djidjev and Shankar M. Venkatesan, February 1991
TR-187	**Multisearch Techniques for Implementing Data Structures on a Mesh-Connected Computer** Mikhail J. Atallah, Frank Dehne, Russ Miller, Andrew Rau-Chaplin, and Jyh-Jong Tsay, February 1991
TR-188	**Generating and Sorting Jordan Sequences** Alan Knight and Jörg-Rüdiger Sack, March 1991
TR-189	**Probabilistic Estimation of Damage from Fire Spread** Charles C. Colbourn, Louis D. Nel, T.B. Boffey and D.F. Yates, April 1991
TR-190	**Coordinators: A Mechanism for Monitoring and Controlling Interactions Between Groups of Objects** Wilf R. LaLonde, Paul White, and Kevin McGuire, April 1991
TR-191	**Towards Decomposable, Reusable Smalltalk Windows** Kevin McGuire, Paul White, and Wilf R. LaLonde, April 1991
TR-192	**PARASOL: A Simulator for Distributed and/or Parallel Systems** John E. Neilson, May 1991
TR-193	**Realizing a Spatial Topological Data Model in a Relational Database Management System** Ekow J. Otoo and M.M. Allam, August 1991
TR-194	**String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations** B John Oommen, September 1991
TR-195	**The Expressiveness of Silence: Optimal Algorithms for Synchronous Communication of Information** Una-May O'Reilly and Nicola Santoro, October 1991
TR-196	**Lights, Walls and Bricks** J. Czyzowicz, E. Rivera-Campo, N. Santoro, J. Urrutia and J. Zaks, October 1991
TR-197	**A Brief Survey of Art Gallery Problems in Integer Lattice Systems** Evangelos Kranakis and Michel Pocchiola, November 1991
TR-198	**On Reconfigurability of Systolic Arrays** Amiya Nayak, Nicola Santoro, and Richard Tan, November 1991
TR-199	**Constrained Tree Editing** B. John Oommen and William Lee, December 1991
TR-200	**Industry and Academic Links in Local Economic Development: A Tale of Two Cities** Helen Lawton Smith and Michael Atkinson, January 1992
TR-201	**Computational Geometry on Analog Neural Circuits** Frank Dehne, Boris Flach, Jörg-Rüdiger Sack, Natana Valiveti, January 1992
TR-202	**Efficient Construction of Catastrophic Patterns for VLSI Reconfigurable Arrays** Amiya Nayak, Linda Pagli, Nicola Santoro, February 1992
TR-203	**Numeric Similarity and Dissimilarity Measures Between Two Trees** B. J. Oommen, K. Zhang and W. Lee, February 1992 (Revised July 1993)