A Common Basis for Similarity Measures Involving Two Strings

R.L. Kashyap* and B.J. Oommen**

SCS-TR-14
January 1983

* Department of Electrical Engineering, Purdue University, W. Lafayette, IN, 47907, USA.

** School of Computer Science, Carleton University, Ottawa, Ontario, K1S 5B6, Canada.

To be published in the International Journal of Computer Mathematics, Vol. 13, No. 1, March 1983.

A COMMON BASIS FOR SIMILARITY MEASURES INVOLVING TWO STRINGS[†]

Kashyap, R.L., and Oommen, B.J.,

ABSTRACT

Many numerical indices which quantify the similarity and dissimilarity between a pair of strings, X and Y, have been defined in the literature. Some of these include the Length of their Longest Common Subsequence (LLCS(X,Y)), the Length of their Shortest Common Supersequence (LSCS(X,Y)), and their Generalized Levenshtein Distance (GLD(X,Y)). Some non-numerical indices relating the strings are the set of their common subsequences, the set of their common supersequences and the set of their shuffles. In this paper, we consider an abstract measure between X and Y, written as D(X,Y), defined in terms of two abstract operators $\oplus$ and $\otimes$ and a binary function $d(\cdot,\cdot)$ whose arguments are symbols of an alphabet $\tilde{A}$. Depending on the various concrete operators used for $\oplus$ and $\otimes$ and the specific function used for $d(\cdot,\cdot)$, all the quantities discussed above can be seen to be particular cases of D(X,Y). We have presented an algorithm to recursively compute D(X,Y), which can serve to be a common scheme to compute all these quantities. Many new results are obtained using this abstract formulation, such as an explicit linear relationship between the LLCS and the LSCS between two strings. It can also be seen that the algorithm to compute the LLCS between X and Y, is a special case of the algorithm to compute D(X,Y). In addition, by using different concrete values for $\oplus$, $\otimes$ and $d(\cdot,\cdot)$, new one-pass algorithms can be developed to compute various quantities such as the set of

[†]Partially supported by the National Science Foundation Grant No. ECS-80-09041.

Longest Common Subsequences (LCS), the set of all the Shortest Common Supersequences (SCS) without backtracking, and the set of all the shuffles of two strings.

Index Terms: Similarity measures for strings, common basis for properties involving strings, distances between strings, longest common subsequence, shortest common supersequence, sets of shuffles of two strings.

I. Introduction

Consider a pair of strings X and Y, made up of symbols from a finite alphabet, A. There are many quantifications of the similarity and dissimilarity between them. Some of them, like the Generalized Levenshtein Distance (GLD), [6, 7, 10, 12, 13, 14, 15], the Length of their Longest Common Subsequence (LLCS) [1, 3, 4, 5, 6, 10, 11] and the Length of their Shortest Common Supersequence (LSCS) [9] are numerical indices. Others, like the set of their common subsequences, the set of their common supersequences, the set of their LCS etc., are nonnumerical quantities. Currently, there seems to be no common basis relating these various measures of similarity. Indeed, the computational algorithm to compute the LLCS bears a marked resemblance [10, 14] to the algorithm that computes the GLD. But the actual underlying reason for this resemblance between the properties themselves (as opposed to the resemblance between the algorithms computing them) has not been investigated.

In this paper we shall define an abstract measure of comparison, D(X,Y) between X and Y in terms of a set of elementary measures, d(·,·) and an algebraic structure τ including two operators $\oplus$ and $\otimes$. By appropriately choosing the two operators and d(·,·), all the measures of similarity and dissimilarity defined above can be obtained as special cases. Other quantities defined over a pair of strings X and Y, such as the set of their shuffles, and P(Y/X), the probability of receiving Y given that X was transmitted across a channel causing independent substitution and deletion errors, can also be regarded as special cases of the abstract measure.

We also show that D(X,Y) can be recursively computed, i.e., D(X,Y) can be computed using $D(\underline{X}, Y)$, $D(X,\underline{Y})$ and $D(\underline{X},\underline{Y})$, where $\underline{Z}$ is the prefix of a string Z of length R-1, if Z is of length R. Thus the same algorithm with the appropriate choice of the operators and of d(·,·) will compute all of the above measures, whether numerical or not. This algorithm can also be used to compute the *set of all* the LCS between two strings without performing *any* backtracking, which no existing algorithm performs.

We list below some of the specific new contributions of this paper.

(1) An explicit relationship between LLCS(X,Y) and LSCS(X,Y), so that all the fast algorithms for computing LLCS(X,Y) can be used equally efficiently to compute the LSCS(X,Y).

(2) An algorithm to compute the set of all the shuffles of two strings.

(3) An algorithm to compute the set of all the LCS between two strings, *without backtracking*.

(4) An algorithm to compute the set of all the SCS between two strings, *without backtracking*.

After reviewing the notation in this section, we proceed to define the abstract measure D(X,Y) in the next. In Section 3 we study two numerical quantities which are special cases of D(X,Y), namely GLD(X,Y) and P(Y/X). In Section 4 we study all the properties of subsequences, supersequences and shuffles of a pair of strings that are captured by D(X,Y). The computational aspects of D(X,Y) are analysed in section 5 and an algorithm, referred to as Algorithm I, presented, to compute it. We extend the results obtained in this paper to study similarity measures for sets of strings in [6].

I.1. Notation

Let A be a finite alphabet and A^* be the set of strings over A. μ, the null string, is distinct from λ, the null symbol. Let $\tilde{A} = A \cup \{\lambda\}$. $\tilde{A}$ is referred to as the Appended Alphabet. A string $X \in A^*$ of the form $X = x_1 \ldots x_N$, where each $x_i \in A$, is said to be of length $|X| = N$. Its prefix of length i will be written as $X_i, i \leq N$. Upper case symbols represent strings and lower case symbols, elements of the alphabet under consideration.

The set concatenation operator "$\cdot$" is defined as follows. Let Q_1 and Q_2 be subsets of A^*. Then,

$$Q_1 \cdot Q_2 = \{XY | X \in Q_1, Y \in Q_2\} \text{ ; XY is the string concatenation of X and Y.}$$

Let Z' be any element in $\tilde{A}^*$, the set of strings over $\tilde{A}$. The Compression Operator, C, is a mapping from $\tilde{A}^*$ to A^*. C(Z') is Z' with all the occurences of the symbol λ removed. Z'. Note that C preserves the order of the non-λ symbols in Z'. For example, if $Z' = f\lambda o\lambda r$, C(Z') = for.

The sets Z_p and R_p are the sets of nonnegative integers and real numbers respectively. Let $R_p' = R_p \cup \{\infty\}$ and $Z_p' = Z_p \cup \{\infty\}$.

<u>II. An Abstract Measure for the Comparison of Two Strings</u>

We consider a functional D(X,Y), between two strings X and Y, $X, Y \in A^*$, induced by a system (A,τ,d), where A is the alphabet under consideration, τ is an algebraic structure and d is a binary function over $\tilde{A} \times \tilde{A}$. We explicitly define τ and d below.

II.1. The Abstract Structure τ.

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

(i) $(T, \oplus, \theta)$ is a commutative monoid [8], 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 $\oplus$. Further, $\theta \in T$ is the identity for $\oplus$.

(ii) $(T, \otimes, I)$ is a monoid with $I \in T$ as the identity for $\otimes$. We emphasize that $\otimes$ need not be commutative.

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

Any algebraic structure satisfying the above conditions is termed a well-defined structure.

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

$$\underset{i=1}{\overset{n}{\textcircled{\Sigma}}} h_i \triangleq h_1 \oplus \cdots \oplus h_n$$

$$\underset{i=1}{\overset{n}{\textcircled{\Pi}}} h_i \triangleq ((\cdots(h_1 \otimes h_2) \otimes h_3) \cdots h_n)$$

Since $\oplus$ is commutative, the order in which the operations are performed in the computation of $\textcircled{\Sigma}$ is of no consequence. Since $\otimes$ need not be commutative, we follow the convention that the operations performed in the computation of $\textcircled{\Pi}$ are performed from left to right, as specified by the parenthesis, though of course, the computation can be performed in any sequence consistent with the ordering.

<u>Example 1</u>. Consider the structure $\tau_1 = (R_p^{'}, \text{MIN}, +, \infty, 0)$, where

(i) $R_p^{'}$ is the set consisting of the nonnegative real numbers and ∞.

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

$$\begin{aligned} \text{MIN}(p,q) &= p, \text{ if } p \leq q \\ &= q \text{ otherwise} \end{aligned}$$

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.

<u>Example 2</u>. Another algebraic structure used in this paper is τ_2:

$$\tau_2 = (R_p', +, *, 0, 1), \text{ where}$$

+ and * represent arithmetic addition and multiplication respectively. Clearly $(R_p', +, 0)$ is a commutative monoid and $(R_p', *, 1)$ is a monoid. Further since * distributes over + from both sides, τ_2 is a well-defined algebraic structure.

<u>Example 3</u>. Let $\tau_4 = (T_A, \cup, \cdot, \emptyset, \{\mu\})$, where

(i) T_A is the power set of A^*, the set of strings over A.

(ii) $\cup$ and $\cdot$ are the set union and set concatenation operators respectively, with identities $\emptyset$ and $\{\mu\}$ respectively.

Since $(T_A, \cup, \emptyset)$ is an associative and commutative monoid, $(T_A, \cdot, \{\mu\})$ an associative monoid, and since $\cdot$ distributes over $\cup$ from both sides, τ_4 is also a well defined structure.

The generalized Warshall's Algorithm described in [2 (Sections 5.6 and 5.7)] for graph theoretic applications also uses operators like $\oplus$ and $\otimes$. However, the operators defined in τ are less restrictive than those defined in [2] as indicated below.

(i) θ need not be the annihilator for $\otimes$, i.e., for any $h_1 \in \tau$, $h_1 \otimes \theta$ need not necessarily be θ.

(ii) $\oplus$ need not be idempotent, i.e., for any $h_1 \in \tau$, $h_1 \oplus h_1$ need not necessarily be equal to n_1.

(iii) $\otimes$ need not distribute over $\oplus$ for countably infinite applications of $\oplus$.

For example, the operators in [2] cannot be used to represent arithmetic addition and multiplication as in structure τ_2.

II.2. The Set of Elementary Measures: d

$d(\cdot,\cdot)$ is a function whose arguments are a pair of symbols belonging to $\tilde{A}$, the appended alphabet, and whose range is T, the set defined in τ. $d(\Lambda,\Lambda)$ is undefined all through and is not needed. The elementary measure $d(a,b)$ can be interpreted as the measure associated with transforming 'b' to 'a', for $a, b \in \tilde{A}$.

The function $d(\cdot,\cdot)$ with the set $R_p^{'}$ as its range has been used earlier to compute the Generalized Levenshtein Distance (GLD) between two strings [6, 7, 10, 13, 14, 15], and to compute the Length of the Longest Common Subsequence (LLCS) between them [3, 4, 10, 11]. However, $d(\cdot,\cdot)$ can assume nonnumerical values too.

Example 4. Consider the structure τ_4 of Example 3. One possible function that can be defined from $\tilde{A} \times \tilde{A}$ to T_A is d_4.

$$
\begin{aligned}
d_4(a,b) &= \{a\}, \quad \text{if } a = b \neq \Lambda; a,b \in \tilde{A} \\
&= \{\mu\}, \quad \text{otherwise}
\end{aligned}
$$

The system (A,τ_4,d_4) can be used to generate the set of all common subsequences between two strings.

II.3. The Set $G_{X,Y}$

For every pair (X,Y), $X,Y \in A^*$, the finite set $G_{X,Y}$ is defined by means of the compression operator C, as a subset of $\tilde{A}^* \times \tilde{A}^*$.

$$G_{X,Y} = \{(X',Y') \mid (X',Y') \in \tilde{A}^* \times \tilde{A}^*, \text{ and obeys (i) - (iii)}\} \quad (1)$$

(i) $C(X') = X$, $C(Y') = Y$

(ii) $|X'| = |Y'|$

(iii) In no (X',Y') is $x_i' = y_i' = \lambda$, $1 \leq i \leq |X'|$.

By definition, if $(X',Y') \in G_{X,Y}$, $\text{Max}\,[|X|, |Y|] \leq |X'| = |Y'| \leq |X| + |Y|$.

The meaning of the pair $(X',Y') \in G_{X,Y}$ is that it corresponds to one way of editing Y into X, using the edit operations of substitution, deletion and insertion. The edit operations themselves are specified for all $i=1,\ldots,|X'|$ by (x_i',y_i'), which represents the transformation of y_i' to x_i'. The cases below consider the three edit operations individually.

(i) If $x_i' \in A$ and $y_i' \in A$, it represents the substitution of y_i' by x_i'.

(ii) If $x_i' \in A$ and $y_i' = \lambda$, it represents the insertion of x_i'. Between these two cases, all the symbols in X are accounted for.

(iii) If $x_i' = \lambda$ and $y_i' \in A$, it represents the deletion of y_i'. Between cases (i) and (iii) all the symbols in Y are accounted for.

$G_{X,Y}$ is an exhaustive enumeration of the set of all the ways by which Y can be edited to X using the edit operations of substitution, insertion and deletion without destroying the order of the occurence of the symbols in X and Y.

The number of elements in the set $G_{X,Y}$ is given by

$$|G_{X,Y}| = \sum_{k=\max[0,|Y|-|X|]}^{|Y|} \frac{(|X|+k)!}{k!\,(|Y|-k)!\,(|X|-|Y|+k)!}$$

Note that $|G_{X,Y}|$ depends only on $|X|$ and $|Y|$, and not on the actual strings X and Y themselves.

<u>Example 5</u>. Let $X = f$ and $Y = go$. Then,

$$G_{X,Y} = \{(f\lambda,go),\ (\lambda f,go),\ (f\lambda\lambda,\lambda go),(\lambda f\lambda,g\lambda o),\ (\lambda\lambda f,go\lambda)\}$$

In particular the pair $(\lambda f,go)$ represents the edit operations of deleting the 'g' and replacing the 'o' by an 'f'.

II.4. The Abstract Measure D(X,Y)

The Abstract Measure $D(X,Y)$ between $X,Y \in A^*$, induced by the system (A,τ,d) where $\tau = (T, \oplus, \otimes, \theta, I)$, is a map whose domain is $A^* \times A^*$ and whose range is T. Formally,

$$D(X,Y) = I, \text{ the identity for } \otimes, \quad \text{if } X = Y = \mu.$$

$$= \underset{(X',Y') \in G_{X,Y}}{\textcircled{\scriptsize Σ}} \left[\overset{|X'|}{\underset{i=1}{\textcircled{\scriptsize Π}}}\ d(x_i',y_i') \right] \quad \text{otherwise.} \qquad (2)$$

The fact that $D(X,Y)$ is well-defined follows directly from the associativity of $\otimes$ and the associativity and commutativity of $\oplus$. Since the measure $D(X,Y)$ is defined in terms of the set $G_{X,Y}$, it can be used to define a number of numerical and nonnumerical indices relating X and Y.

In general, $D(X,Y)$ need not be equal to $D(Y,X)$. $D(X,Y)$ will be symmetric if and only if the function $d(\cdot,\cdot)$ inducing it obeys the following conditions:

$$d(a,b) = d(b,a) \qquad \text{for all } a,b \in A$$

$$d(\lambda,a) = d(a,\lambda) \qquad \text{for all } a \in A.$$

<u>III. Numerical Measures D(X,Y)</u>

III.1. The Generalized Levenshtein Distance

The <u>G</u>eneralized <u>L</u>evenshtein <u>D</u>istance (GLD) between two strings X and Y (written as GLD (X,Y)) is defined as the minimum of the sum of the elementary edit distances associated with the edit operations required to transform Y to X [6, 7, 10, 13, 14, 15]. The elementary edit distances themselves are specified in terms of a map $d_1(\cdot,\cdot)$ from $\tilde{A} \times \tilde{A}$ to R_p', the set consisting of the nonnegative real numbers and ∞. $d_1(\cdot,\cdot)$ obeys the following conditions:

$$\begin{aligned} d_1(a,b) &> 0 \quad \text{for all} \quad a \neq b,\ a,b \in \tilde{A} \\ &= 0 \quad \text{if } a = b, \quad a,b \in A. \\ d_1(a,b) &\leq d_1(a,c) + d_1(c,b) \text{ for all } a,b \in A,\ c \in \tilde{A} \end{aligned}$$

Subject to the above constraints, the GLD obeys the following for all $X,Y,Z \in A^*$ [12].

$$\begin{aligned} GLD(X,Y) &> 0 \qquad \text{if } X \neq Y \\ &= 0 \quad \text{only if } X = Y \\ GLD(X,Y) &\leq GLD(X,Z) + GLD(Z,Y) \end{aligned}$$

In general, a greater value of the GLD between two strings indicates a greater dissimilarity between them.

<u>Theorem 1</u>.

The GLD between X and Y is exactly the measure between them induced by the system (A,τ_1,d_1), where τ_1 is the structure defined in Example 1.

<u>Proof</u>. The theorem is trivially true when $X = Y = \mu$, since $GLD(\mu,\mu) \triangleq 0$. Consider the case when either X or Y or both is not μ. We have already seen that the set of ways by which Y can be edited to X is given by the set $G_{X,Y}$. Any pair $(X',Y') \in G_{X,Y}$, where $X' = x_1'x_2'\ldots$, and $Y' = y_1'y_2'\ldots$ represents the edit operation of transforming y_i' to x_i' for all $1 \leq i \leq |X'|$. Hence the sum of the edit distances corresponding to this pair is given by the number $\sum_{i=1}^{|X'|} d_1(x_i',y_i')$. By definition, the minimum of this quantity is the GLD between X and Y. Since $G_{X,Y}$ is the set of all pairs (X',Y'):

$$GLD(X,Y) = \underset{(X',Y') \in G_{X,Y}}{\text{Minimum}} \left[\sum_{i=1}^{|X'|} d_1(x_i', y_i') \right]$$

which is exactly the value of the measure induced between X and Y by the system (A,τ_1,d_1). Hence the theorem.

***

III.2. Probability of Errorenous Strings

Consider a channel which causes only independent substitution and deletion errors. Let the elementary error probabilities be given by the function mapping $\tilde{A} \times \tilde{A}$ to R_p' as below.

(i) p(b/a) is the conditional probability of receiving the symbol 'b' given that the symbol 'a' is transmitted.

(ii) $p(\lambda/a)$ is the conditional probability of the channel deleting the transmitted symbol 'a'.

Since a transmitted symbol can either be transmitted correctly, deleted or transformed into another symbol, we have,

$$\sum_{v \in \tilde{A}} p(v/a) = 1 \quad \text{for all } a \in A. \tag{3}$$

We define $p(a/\lambda) \triangleq 0$ for all $a \in A$. The quantity $p(\lambda/\lambda)$ is undefined and is not needed.

Let X be any string transmitted across the channel. Let Y be the corresponding noisy received string. Let $|X| = N$. Since the channel cannot cause insertion errors, $|Y| \leq N$. Explicitly $Y \in A_N^*$, where,

$$A_N^* = \{Y | Y \in A^*, |Y| \leq N\}$$

A_N^* is the set of strings over A of length less than or equal to N.

Let (X', Y') be any arbitrary element of $G_{X,Y}$ and let

$$W_2' = \prod_{i=1}^{|X'|} p(y_i'/x_i') \tag{4}$$

where Π represents arithmetic multiplication.

W_2' is the product of the probabilities associated with the pair (X', Y') given that every y_i' was caused by the corresponding x_i'. Since $p(a/\lambda) = 0$ for all $a \in A$, W_2' is identically zero whenever the pair (X', Y') represents the insertion of at least one symbol in Y.

We define P(Y/X) as the likelihood of receiving a Y given that a X is transmitted. P(Y/X) is defined as unity if both X and Y are μ. Otherwise, it is defined as the sum of W_2' over all pairs (X', Y') in $G_{X,Y}$. That is,

$$P(Y/X) = \sum_{(X', Y') \in G_{X,Y}} \prod_{i=1}^{|X'|} p(y_i'/x_i') \tag{5}$$

The quantity P(Y/X) has the following properties.

<u>Theorem 2.</u>

Let τ_2 be the algebraic structure defined in Example 2. P(Y/X), the likelihood of receiving a Y when X is transmitted, is exactly the measure between X and Y induced by the system (A, τ_2, d_2), where $d_2(\cdot,\cdot)$ is defined by:

$$d_2(a,b) \triangleq p(b/a) \quad \text{for all } a,b \in \tilde{A}.$$

<u>Proof.</u> The theorem follows directly by applying (2) to the system (A, τ_2, d_2) and comparing it with (5).

***

Theorem 3.

The likelihood $P(Y/X)$ is a valid probability assignment for the probability of receiving Y, given X is transmitted.

Proof. Since $P(Y/X) \geq 0$, all that has to be proved is that

$$\sum_{Y \in A_N^*} P(Y/X) = 1 \tag{6}$$

Since the channel cannot cause insertion errors, $(X', Y') \in G_{X,Y}$ will yield a nonzero contribution to $P(Y/X)$ iff $|X'| = |Y'| = N$. Let the subset of elements of $G_{X,Y}$ which have $|X'| = |Y'| = N$ be G_N. Then,

$$P(Y/X) = \sum_{(X',Y') \in G_N} \prod_{i=1}^{N} p(y_i'/x_i'), \quad \text{where } x_i' = x_i,\ i = 1,\ldots,N, \tag{7}$$

The sum on the RHS of (7) involves independently considering <u>all</u> the possible transformations of the symbols of X. Thus,

$$\sum_{Y \in A_N^*} P(Y/X) = \prod_{i=1}^{N} \sum_{v \in \tilde{A}} p(v/x_i)$$

which equals unity due to (3).

***

Remark:

The fact that we have not required $\oplus$ to be idempotent as in [2], has permitted us to let $\oplus$ and $\otimes$ represent the arithmetic addition and multiplication respectively.

IV. Subsequences, Supersequences and Shuffles

The measure D(X,Y) induced by the system (A,τ,d) captures all the properties involving the subsequences, supersequences and shuffles of a pair of strings. We study these properties in this section.

IV.1. The Length of the Longest Common Subsequence

A string $Z \in A^*$ is a subsequence of X if it can be obtained from X by deleting some of the symbols (not necessarily contiguous) of X. A common subsequence of two strings X and Y of maximum length is defined as one of their Longest Common Subsequences (LCS) and its length is referred to as the Length of the Longest Common Subsequence (LLCS) [1, 3, 4, 5, 9, 10, 11, 13, 14]. Determining the LLCS of two strings has applications in molecular biology [1, 11] and in data processing [1].

Most of the research that has involved the LCS of two strings, has been related to efficiently computing the length of their LCS. Hirschberg proposed a linear space algorithm to compute the LLCS between two strings, and any one of their LCS [4]. He later proposed [4] two fast algorithms to compute the LLCS of two strings, and in the special case when the LLCS was of the same order as the length of the strings, the second algorithm in [4] required time that was much less than quadratic. Hunt and Szymanski [5] proposed an algorithm to compute the LLCS which had a worst case complexity which was of $O(n^2 \log n)$, where n is the length of the strings. However, for strings in which most positions in one sequence match relatively few positions in the other, the algorithm requires only $O(n \log n)$ time. Bounds on the complexity of the LCS problem have also been derived [1, 15]. Although the LCS problem has been extensively studied, there has been no algorithm presented to compute the set of all the LCS between two strings. Further, there has been no algorithm proposed to even compute the LLCS of more than two strings. In this paper and its companion [6], we shall present algorithms which compute these quantities.

We shall refer to the LLCS between two strings X and Y as LLCS(X,Y). By definition if $X = Y = \mu$, LLCS(X,Y) = 0. Consider the nontrivial case when either X or Y (or both) is not μ.

Let $d_3(\cdot,\cdot)$ be a function defined from $\tilde{A} \times \tilde{A}$ to Z_p' by:

$$\begin{aligned} d_3(a,b) &= 1 \qquad \text{if } a = b \neq \lambda \quad a,b \in \tilde{A} \\ &= 0 \qquad \text{otherwise} \end{aligned}$$

Consider any pair $(X',Y') \in G_{X,Y}$. Corresponding to this pair we can associate an integer W_3', where,

$$W_3' = \sum_{i=1}^{|X'|} d_3(x_i',y_i').$$

By the definition of $d_3(\cdot,\cdot)$, the value of W_3' is exactly the number of non-λ pairs of symbols (x_i',y_i') in (X',Y') with $x_i' = y_i'$, which if concatenated from left to right will yield a common subsequence between X and Y. Hence W_3' is the length of <u>a</u> common subsequence. By the definition of $G_{X,Y}$, the maximum value of W_3' is the value LLCS(X,Y). Hence,

$$\mathrm{LLCS}(X,Y) = \underset{(X',Y') \in G_{X,Y}}{\text{Maximum}} \left[\sum_{i=1}^{|X'|} d_3(x_i',y_i') \right] \tag{8}$$

This leads to the following theorem.

<u>Theorem 4.</u>

Let τ_3 be the abstract structure $(Z_p', \text{MAX}, +, 0, 0)$, where

(i) Z_p' is the set consisting of the nonnegative integers and ∞.

(ii) MAX is the commutative binary operator defined by

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

(iii) + represents arithmetic addition

Then the LLCS(X,Y) is the measure induced by the system (A,τ_3,d_3).

<u>Proof</u>. The theorem follows by applying (2) to the system (A,τ_3,d_3) and comparing the result with (8).

***

IV.2. The Set of Common Subsequences

A similar result can be obtained for the set of common subsequences between X and Y. Let $d_4(\cdot,\cdot)$ be a function defined as in Example 4. Consider two strings, X and Y, where either or both X and Y are not null. For any pair $(X',Y') \in G_{X,Y}$, let W_4' be the set given by:

$$W_4' = \prod_{c\ i=1}^{|X'|} d_4(x_i',y_i')$$

where Π_c represents the iterative use of '$\cdot$', the set concatenation operator.

By virtue of the elementary measures, W_4' is a singleton set containing <u>a</u> common subsequence of X and Y. Further, since $G_{X,Y}$ is an exhaustive enumeration of all the pairs (X',Y'), the union of W_4' over all pairs (X',Y') is the set of all the common subsequences between X and Y, and is given by (9).

$$\bigcup_{(X',Y') \in G_{X,Y}} \left[\prod_{c\ i=1}^{|X'|} d_4(x_i',y_i') \right] \tag{9}$$

The latter is exactly the measure induced between them by the system

(A,τ_4,d_4). Hence we have the following theorem.

Theorem 5.

Let τ_4 and d_4 be as defined in Examples 3 and 4 respectively. The set of common subsequences between X and Y is exactly the measure, $D^4(X,Y)$, induced between them by the system (A,τ_4,d_4).

***

Remark: The set of all the LCS of X and Y can be obtained by extracting from $D^4(X,Y)$ the elements of greatest length.

IV.3. The Shortest Common Supersequence Problem

An analogous problem to the LCS problem is the Shortest Common Supersequence (SCS) Problem [9]. Z is a supersequence of X if it can be obtained from X by inserting symbols into it. A common supersequence of X and Y of minimum length is one of their Shortest Common Supersequences (SCS), and its length is the Length of the Shortest Supersequence of the strings X and Y, written as LSCS(X,Y).

The next two theorems concern the SCS problem. The arguments proving the theorems follow the same reasoning as the previous two and are omitted here to avoid needless repetition.

Theorem 6.

Let τ_5 be the structure $(Z_p^{'},\mathrm{MIN},+,\infty,0)$. Let $d_5(\cdot,\cdot)$ be a function which specifies a set of elementary measures. $d_5(\cdot,\cdot)$ maps $\tilde{A} \times \tilde{A}$ to $Z_p^{'}$ by:

$$d_5(a,b) = 1 \qquad \text{if } a = b, \quad a,b \in A$$
$$= \infty \qquad \text{otherwise}$$
$$d_5(a,\lambda) = d_5(\lambda,a) = 1 \quad \text{for all } a \in A.$$

Then $D^5(X,Y)$, the measure induced between X and Y by (A,τ_5, d_5), is exactly equal to LSCS(X,Y).

***

Theorem 7.

Let τ_4 be the algebraic structure defined in Example 3. Further, let $d_6(\cdot,\cdot)$ be a function from $\tilde{A} \times \tilde{A}$ to the power set of A^* defined by:

$$\begin{aligned} d_6(a,b) &= \{a\} && \text{if } a = b, \quad a,b \in A \\ &= \emptyset && \text{otherwise} \\ d_6(\lambda,a) = d_6(a,\lambda) &= \{a\} && a \in A. \end{aligned}$$

Then $D^6(X,Y)$, the measure between X and Y induced by (A,τ_4, d_6), is the set of their common supersequences of length less than or equal to $|X|+|Y|$.

***

Comments

(i) The set of all the SCS of X and Y can be obtained by extracting from $D^6(X,Y)$ the strings of smallest length.

(ii) If we follow through the results of the the above four theorems, we notice a marked similarity between the LCS and the SCS problem. A closer examination will reveal that if we are given X, Y and any one of their LCS, we can obtain at least one of their SCS. An SCS is generated by including in it every symbol not in the LCS, and by including every symbol in the LCS exactly once. Thus we obtain the following result relating the quantities LLCS(X,Y) and LSCS(X,Y).

Corollary 1.

Let $X,Y \in A^*$. Then,

$$\mathrm{LLCS}(X,Y) = |X| + |Y| - \mathrm{LSCS}(X,Y)$$

Proof.

Let $X=x_1 \ldots x_N$ and $Y=y_1 \ldots y_M$, and let $Z=z_1 \ldots z_R$ be any LCS of X and Y. Then, by definition,

$$x_{i_k} = z_k \text{ for } k=1,\ldots,R \text{ with } i_{p-1} < i_p, p=2,\ldots,R, \text{ and,}$$

$$y_{j_k} = z_k \text{ for } k=1,\ldots,R \text{ with } j_{p-1} < j_p, p=2,\ldots,R.$$

We edit X using the edit operation of insertion as below. Examine X starting from the left. When the symbol x_{i_k} in the LCS is encountered, insert before it $y_{j_{k-1}}$ if $j_{k-1}>0$. Finally, concatenate to X the substring $y_{j_R+1} \ldots y_M$. Let the resulting string be V. Clearly V is a common supersequence of both X and Y since both of them can be obtained by editing V using only the edit operation of deletion.

V is a Shortest common supersequence of the strings, since, removing any symbol from it will not result in a supersequence. This can be easily seen by considering all the possibilities of deleting in V one symbol either from X or from Y.

By the construction,

$$|V| = |X| + (|Y| - |Z|), \quad |V|=\mathrm{LSCS}(X,Y), \ |Z|=\mathrm{LLCS}(X,Y).$$

IV.4. The Set of Shuffles

A string $Z \in A^*$ is referred to as a shuffle of X and Y if it satisfies the following conditions:

a) Z is a common supersequence of both X and Y.

b) There exists some way of deleting the symbols of X from Z to leave the remnant string exactly Y, and vice versa.

The following theorem shows that the set of shuffles of the strings X and Y is the measure induced between them by the system (A,τ_4,d_7) described below.

Theorem 8.

Let τ_4 be the algebraic structure defined in Example 3. Let $d_7(\cdot,\cdot)$ be a function mapping $\tilde{A} \times \tilde{A}$ to the power set of A^*, defined by:

$$d_7(a,b) = \emptyset \qquad \text{for all } a,b \in A$$
$$d_7(a,\lambda) = d_7(\lambda,a) = \{a\} \qquad \text{for all } a \in A.$$

Then $D^7(X,Y)$, the measure between X and Y induced by the system (A,τ_4,d_7) is the set of all the shuffles of X and Y.

Proof. The proof of the result follows directly from the fact that the contribution of every pair (X',Y') to $D^7(X,Y)$ is a singleton set which, by virtue of the elementary measures, is a shuffle of X and Y. Since $G_{X,Y}$ is the exhaustive collection of all the pairs (X',Y'), for every shuffle Z, there is some pair $(X',Y') \in G_{X,Y}$ which yields its contribution to $D^7(X,Y)$ as Z itself. Thus, the union of all these singleton sets will be the set of all the shuffles of X and Y. Hence the theorem.

***

V. Computational Properties of D(X,Y)

Let $\underline{Z}$ be defined as the left derivative of any string $Z \in A^*$, so that $\underline{Z} = z_1 \ldots z_{R-1}$ if $Z = z_1 \ldots z_R$. By definition $\underline{Z} = \mu$ if $|Z| \leq 1$. In this section we show that D(X,Y) is explicitly dependent only on the quantities $D(\underline{X},Y)$, $D(Y,\underline{X})$, $D(\underline{X},\underline{Y})$, and the elementary measures $d(\cdot,\cdot)$ involving the last symbols of X and Y. This result is primarily due to the properties of the operators $\oplus$ and $\otimes$ and the way they appear in the expression for D(X,Y).

This property of $D(X,Y)$ renders it recursively computable and leads to the interesting and computationally efficient feature that though $D(X,Y)$ is defined explicitly in terms of the set $G_{X,Y}$, it can be computed recursively without ever computing $G_{X,Y}$. Thus, though the set $G_{X,Y}$, whose size is given in section 3, is extremely large, the measure $D(X,Y)$ can be computed using merely $O(|X||Y|)$ computations involving the operators $\otimes$ and $\oplus$.

We shall first prove the recursive computatability of $D(X,Y)$, and present an algorithm to compute it. The algorithm is written in terms of the system (A,τ,d). By using various concrete systems, the same algorithm can be employed to compute all the measures discussed in the previous two sections, such as the $GLD(X,Y)$, $P(Y/X)$, $LLCS(X,Y)$, $LSCS(X,Y)$, and the sets of common subsequences, supersequences and shuffles of X and Y.

We make use of the following notation that if $Z = z_1 \ldots z_R$, $Z \in A^*$, then Z_i is the prefix of Z of length i. By definition, $Z_0 = \mu$ and $\underline{Z} = Z_{R-1}$.

<u>Theorem 9</u>.

Let $X = x_1 \ldots x_N$ and $Y = y_1 \ldots y_M$, where both X and Y are not null. Then,

a) $D(X,\mu) = D(\underline{X},\mu) \otimes d(x_N,\lambda)$

b) $D(\mu,Y) = D(\mu,\underline{Y}) \otimes d(\lambda,y_M)$

c) $$D(X,Y) = \left[D(\underline{X},\underline{Y}) \otimes d(x_N,y_M)\right] \oplus \left[D(X,\underline{Y}) \otimes d(\lambda,y_M)\right] \oplus \left[D(\underline{X},Y) \otimes d(x_N,\lambda)\right]$$

The proof of the theorem is given in Appendix A.

***

Theorem 9 leads to the following algorithm to compute $D(X,Y)$.

<u>Algorithm</u> <u>I</u>.

<u>Input</u>: Two strings $X = x_1 \ldots x_N$ and $Y = y_1 \ldots y_M$.

<u>Output</u>: The Abstract Measure $D(X,Y)$ induced by (A,τ,d).

<u>Method</u>:

```
$D(\mu,\mu) = I$.
for i=1 to N do $D(X_i,\mu) = D(X_{i-1},\mu) \otimes d(x_i,\lambda)$
for j=1 to M do $D(\mu,Y_j) = D(\mu,Y_{j-1}) \otimes d(\lambda,y_j)$
for i=1 to N do
    for j=1 to M do
        $D(X_i,Y_j) = \left[D(X_{i-1},Y_{j-1}) \otimes d(x_i,y_j)\right] \oplus \left[D(X_i,Y_{j-1}) \otimes d(\lambda,y_j)\right] \oplus \left[D(X_{i-1},Y_j) \otimes d(x_i,\lambda)\right]$
    end
end
return $D(X_N,Y_M)$
END Algorithm I.
```

The algorithm is best understood by means of the trellis shown in Fig. 1. The nodes of the graph correspond to the prefixes of the strings, (X_i,Y_j). There is a directed edge to (X_i,Y_j) from each of the nodes (X_{i-1},Y_j), (X_i,Y_{j-1}) and (X_{i-1},Y_{j-1}). The latter three nodes are referred to as the nodes <u>adjacent</u> to (X_i,Y_j). Associated with the edges of the graph are the following elementary measures:

(i) $d(x_i,\lambda)$ is the measure associated with the edge connecting (X_{i-1},Y_j) to (X_i,Y_j), for all $1 \leq j \leq M$.

(ii) $d(\lambda,y_j)$ is the measure associated with the edge connecting (X_i,Y_{j-1}) to (X_i,Y_j), for all $1 \leq i \leq N$.

(iii) $d(x_i,y_j)$ is the measure associated with the edge connecting (X_{i-1},Y_{j-1}) to (X_i,Y_j).

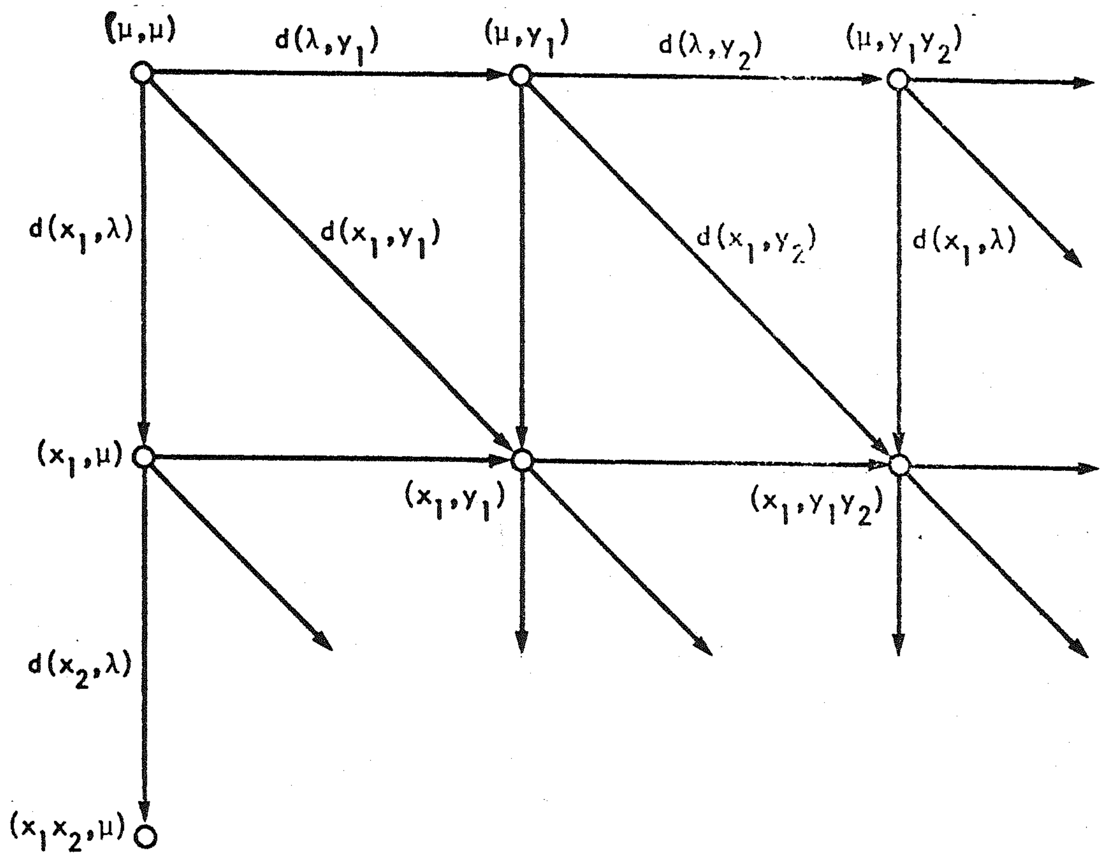

Fig. 1. The trellis to compute D(X,Y). Note that the edges of the trellis correspond to the <u>abstract</u> <u>elementary</u> measures d(.,.). The measure D(X,Y) at a node K is computed via the operators $\circledast$ and $\oplus$ from the measures $D(\underline{X},Y)$, $D(X,\underline{Y})$ and $D(\underline{X},\underline{Y})$ at the tail of the edges leading to the node K.

Associated with the node (X_i, Y_j) is the measure $D(X_i, Y_j)$ which is to be computed. The initialization step makes $D(\mu,\mu) = I$. The X and Y axes are then traced using the edit operations of insertion and deletion respectively. Subsequently, the rest of the trellis is traced, row by row, using only the previously computed measures. We emphasize that to compute the measure associated with any node, we need to use <u>only</u> the measures associated with its adjacent nodes, i.e., (X_{i-1}, Y_j), (X_i, Y_{j-1}) and (X_{i-1}, Y_{j-1}).

V.1. Remarks

(i) As stated earlier, the above algorithm is written in terms of the parameters of the system (A, τ, d). Based on the concrete system used, Algorithm I can be used to compute the various quantities discussed in the previous sections. In particular, it can thus be employed to compute (a) GLD(X,Y), (b) P(Y/X), (c) LLCS(X,Y), (d) LSCS(X,Y), (e) the set of common subsequences of X and Y, (f) the set of common supersequences of X and Y and (g) the set of all the shuffles of X and Y.

(ii) The computational complexity of algorithms similar to Algorithm I has been studied in terms of the number of symbol comparisons required by the algorithm [1, 15]. On this basis, it is easy to see that Algorithm I requires MN symbol comparisons. The algorithm performs the operation $\otimes$ exactly 3MN+M+N times, and the operation $\oplus$ exactly 2MN times. Thus to compute <u>all the numerical quantities</u> discussed in the previous sections, the algorithm has a time complexity of O(MN). To compute nonnumerical quantities that are sets, (such as the set of all the LCS between X and Y), the algorithm will perform the corresponding operations such as the set union and set concatenation O(MN) times. The complexity of the algorithm however, will be a function of the number of elements in the set computed, and not merely a function of the parameters M and N. An asymptotically more efficient algorithm to compute only numerical quantities relating two strings X and Y (specifically, GLD(X,Y) and LLCS(X,Y)) is suggested in [10]. Considering the large amount of preprocessing needed in the latter algorithm, it can be useful only when $|X|$ and $|Y|$ are very large.

(iii) A lower bound for the number of symbol comparisons required to compute the LLCS between two strings has been derived in [1]. From Corollary I, this is also the lower bound for the number of symbol comparisons required to compute the LSCS between two strings. An unanswered question is whether similar lower bounds exist for the other measures defined in this paper.

We shall now illustrate Algorithm I by applying it to compute a nonnumerical quantity.

V.2. The Algorithm for the LCS Problem

We shall show how Algorithm I can be used to compute the set of all LCS between X and Y. We have already shown that the measure induced between X and Y by (A,τ_4,d_4) is the set of all the common subsequences between them. Let this measure be designated as $D^4(X,Y)$.

Let us suppose that in the process of computation we retain in the sets $D^4(X_{i-1},Y_j)$, $D^4(X_i,Y_{j-1})$ and $D^4(X_{i-1},Y_{j-1})$, only the elements of maximum length. Then, by keeping track of the lengths of the elements in these sets we can compute $D^4(X_i,Y_j)$ and retain in it only the elements of maximum length. This too can be done iteratively and thus the set of all LCS between X and Y can be computed iteratively†. In a similar way, the set of

† This can be easily formalized as follows. Let S and T be two sets of strings, where all the elements in S and T are of lengths s and t respectively. We define an operator Ⓤ which operates on sets S and T and to yield the resultant set which is either S or T depending on whether $s>t$ or $s<t$ respectively. If $s=t$, S Ⓤ T is defined as $S \cup T$. Using Ⓤ and the set concatenation operator, the set of all the LCS can be seen to be a particluar case of D(X,Y).

all SCS between X and Y can be computed iteratively.

This algorithm yields the set of all the LCS between X and Y <u>without performing any backtracking</u>. Currently, there is <u>no</u> known algorithm to compute this set recursively.

Using the existing algorithms to compute the LLCS and using backtracking, we can envision the computation of this set of all LCS in two stages. The matrix of the LLCS between the prefixes of X and Y is first computed. Let this matrix be L, whose element $L_{i,j}$ is the LLCS between X_i and Y_j. In this stage pointers are maintained through L, which will later be used to retrieve a LCS. After the entire matrix is computed (i.e., $L_{N,M}$ is computed, if $|X| = N$ and $|Y| = M$), one backtracks through L, from $L_{N,M}$ to $L_{0,0}$ emitting a LCS as a path is traced. However,even if the pointers through L are maintained optimally, it can be seen that the number of distinct paths from $L_{N,M}$ to $L_{0,0}$, is, in general, greater than (and never less than) the number of distinct LCS, implying that more than one path yield the same LCS. Thus, in the process of backtracking, invariably a lot of redundant computations will be performed, because of the inevitable traversing of paths which yield no new LCS.

The algorithm presented earlier involves no backtracking and consequently no redundancy. The example below illustrates the computation of the set of all LCS between X and Y.

<u>Example 6</u>. Let X = atoms and Y = tames, where A = {a,e,m,o,s,t}. The elementary measure $d_4(\cdot,\cdot)$ is used.

$$d_4(u,v) = \{u\} \text{ if } u = v,\ u,v \in A.$$

$=\{\mu\}$, otherwise.

The computation of the set of LCS between X and Y is given in Table 1.

VI. Conclusions

We have presented a unifying theory for similarity properties involving a pair of strings, X and Y, and which require the preservation of the order of the symbols of both the strings. Using this theory we have given explicit expressions and a computational algorithm for (i)GLD(X,Y), (ii) the probability of receiving Y given that X was a string transmitted through a channel causing independent substitution and deletion errors, (iii) LLCS(X,Y), (iv) the set of common subsequences of X and Y, (v) LSCS(X,Y), (vi) the set of common supersequences of X and Y, and (vii) the set of all the shuffles of X and Y.

It is well known that string correction can be done by comparing two strings using the GLD between them[12]. However, due to the results of this paper, we can extend the results of previous researchers to perform string correction using any of the numerical measures discussed in this paper. Besides, a finer comparison between almost similar strings can be made, by using some of the nonnumerical indices we have studied. For example, consider three strings X_1,X_2 and Y. Let us suppose LLCS(X_1,Y)=LLCS(X_2,Y). If the LLCS between two strings is used as a measure of the similarity between them, the two strings X_1, and X_2 will be considered equally similar to Y. But if we consider the number of elements in the set of the LCS between X_1 and Y, and the number of elements in the set of LCS between X_2 and Y, we would have a finer measure of similarity to distingush between X_1 and X_2.

The results that have been obtained for a pair of strings will be extended for sets of strings in the companion paper[6].

	μ	t	a	m	e	s
μ	μ	μ	μ	μ	μ	μ
a	μ	μ	a	a	a	a
t	μ	t	a,t	a,t	a,t	a,t
o	μ	t	a,t	a,t	a,t	a,t
m	μ	t	a,t	am,tm	am,tm	am,tm
s	μ	t	a,t	am,tm	am,tm	ams,tms

Table 1 : The computation of the set of LCS between X=atoms and Y=tames. The entry corresponding to x_i and y_j is the set of LCS between X_i and Y_j. Hence, the set of LCS between 'tames' and 'atoms' is {ams, tms}, the entry in the right corner of the last row.

ACKNOWLEDGEMENTS

The authors would like to thank Professor Michael O'Donnell of the Department of Computer Science at Purdue University, W. Lafayette, for his comments on the manuscript and for suggesting the result of Corollary 1. The authors also would like to thank Professor Glenn Manacher of the Department of Mathematics, Univ. of Illinois, Chicago Circle, for carefully reading the manuscript and offering several suggestions for improvement.

REFERENCES

1. Aho, A.V., Hirschberg, D.S., and Ullman, J.D., "Bounds on the Complexity of the Longest Common Subsequence Problem," J-ACM, Vol. 23, 1976, pp. 1-12.

2. Aho, A.V., Hopcroft, J.E., and Ullman, J.D., "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, Mass. 1974.

3. Hirschberg, D.S., "Algorithms for the Longest Common Subsequence Problem," J-ACM, Vol. 24, 1977, pp. 664-675.

4. Hirschberg, D.S., "A Linear Space Algorithm for Computing Maximal Common Subsequences," C-ACM, Vol. 18, 1975, pp. 341-343.

5. Hunt, J.W., and Szymanski, T. G., "A Fast Algorithm for Computing Longest Common Subsequences," C-ACM, Vol. 20, 1977, pp. 350-353.

6. Kashyap, R.L., and Oommen, B.J., "Similarity Measures for Sets of Strings", To Appear in the Int. Journal of Comp. Math., Vol. 13.

7. Levenshtein, A., "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," Sov. Phy. Dokl., Vol. 10, 1966, pp. 707-710.

8. Mac Lane, S., and Birkhoff, G., "Algebra," The MacMillan Company, New York, 1967.

9. Maier, D., "The Complexity of Some Problems on Subsequences and Supersequences," J-ACM, Vol. 25, 1978, pp. 322-336.

10. Masek, W.J., and Paterson, M.S., "A Faster Algorithm Computing String Edit Distances," J. of Comp. and Syst. Sci., Vol. 20, 1980, pp. 18-31.

11. Needleman, S.B., and Wunsch, C.D., "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," J. Mol. Biol., 1970, pp. 443-453.

12. Okuda, T., Tanaka, E., and Kasai, T., "A Method of Correction of Garbled Words Based on the Levenshtein Metric", IEEE Trans. Comp., C-25, 1976, pp. 172-177.

13. Sellers, P.H., "An Algorithm for the Distance between Two Finite Sequences," J. of Combinatorial Theory, Vol. 16, 1974, pp. 253-258.

14. Wagner, R.A., and Fischer, M.J., "The String to String Correction Problem," J-ACM, Vol. 21, 1974, pp. 168-173.

15. Wong, C.K., and Chandra, A.K., "Bounds for the String Editing Problem," J-ACM, Vol. 23, 1976, pp. 13-16.

APPENDIX

Proof of Theorem 9 - (a)

Since $G_{X,\mu}$ has only one element $(x_1x_2\ldots x_N,\lambda\lambda\ldots\lambda)$, by (2),

$$D(X,\mu) = \underset{i=1}{\overset{N}{\textcircled{\Pi}}} d(x_i,\lambda)$$

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

$$\underset{i=1}{\overset{N-1}{\textcircled{\Pi}}} d(x_i,\lambda) \circledast d(x_N,\lambda)$$

But $\underset{i=1}{\overset{N-1}{\textcircled{\Pi}}} d(x_i,\lambda) \triangleq D(\underline{X},\mu)$. Hence $D(X,\mu) = D(\underline{X},\mu) \circledast d(x_N,\lambda)$, proving (a).

In an exactly similar way (b) can be proved. We proceed to prove (c).

Proof of (c)

Since $|X|, |Y| \geq 1$,

$$D(X,Y) = \underset{(X',Y')\in G_{X,Y}}{\textcircled{\Sigma}} \left[\underset{i=1}{\overset{|X'|}{\textcircled{\Pi}}} d(x_i',y_i') \right] \qquad (A.1)$$

For any arbitrary pair $(X',Y') \in G_{X,Y}$, let p' and q' be the last symbols of X' and Y' respectively. The set $G_{X,Y}$ can be partitioned into three mutually exclusive and exhaustive subsets.

$$G_1 = \{(X',Y') \mid (X',Y') \in G_{X,Y},\ p' \in A,\ q' \in A\}$$

$$G_2 = \{(X',Y') \mid (X',Y') \in G_{X,Y},\ p' = \lambda,\ q' \in A\}$$

$$G_3 = \{(X',Y') \mid (X',Y') \in G_{X,Y},\ p' \in A,\ q' = \lambda\}$$

The above sets are mutually exclusive because of their definition. Since both p', $q' \in A \cup \{\lambda\}$, and since both cannot be λ simultaneously, every element in $G_{X,Y}$ will be in only one of the above sets. Thus G_1, G_2 and G_3 partition $G_{X,Y}$. Using the associativity of $\oplus$,

$$D(X,Y) = \left[\bigoplus_{(X',Y')\in G_1} S'\right] \oplus \left[\bigoplus_{(X',Y')\in G_2} S'\right] \oplus \left[\bigoplus_{(X',Y')\in G_3} S'\right] \qquad (A.2)$$

where $S' = \bigotimes_{i=1}^{|X'|} d(x_i',y_i')$.

Consider the terms of (A.2) separately. Using the associativity of $\otimes$, the first term in (A.2) reduces to,

$$\bigoplus_{(X',Y')\in G_1} \bigotimes_{i=1}^{|X'|} d(x_i',y_i') = \bigoplus_{(X',Y')\in G_1} \left[\bigotimes_{i=1}^{|X'|-1} d(x_i',y_i') \otimes d(x_N,y_M)\right]$$

Using the distributivity of $\otimes$ over $\oplus$ the above expression reduces to,

$$\left[\bigoplus_{(X',Y')\in G_1} \bigotimes_{i=1}^{|X'|-1} d(x_i',y_i')\right] \otimes d(x_N,y_M) \qquad (A.3)$$

Let $\underline{X}$ and $\underline{Y}$ be the left derivatives of X and Y respectively.

$$D(\underline{X},\underline{Y}) = \bigoplus_{(\underline{X}',\underline{Y}')\in G_{\underline{X},\underline{Y}}} \left[\bigotimes_{i=1}^{|X'|} d(\underline{x}_i',\underline{y}_i')\right]$$

where $G_{\underline{X},\underline{Y}}$ is the set defined in (1) for $\underline{X}$ and $\underline{Y}$. For every element $(\underline{X}',\underline{Y}') \in G_{\underline{X},\underline{Y}}$, there is a unique element $(\underline{X}'x_N,\ \underline{Y}'y_M)$ in G_1 and vice versa. Replacing every element in G_1 by the corresponding element in $G_{\underline{X},\underline{Y}}$ (A.3) can be rewritten as:

$$\left[\mathop{\textcircled{Σ}}\limits_{(\underline{X}',\underline{Y}')\in G_{\underline{X},\underline{Y}}} \mathop{\textcircled{Π}}\limits_{i=1}^{|X'|} d(\underline{x}_i',\underline{y}_i')\right] \otimes d(x_N,y_M) = D(\underline{X},\underline{Y}) \otimes d(x_N,y_M). \tag{A.4}$$

For every element in G_2, $p' = \lambda$ and $q' = y_M$. Thus the second term in (A.2) becomes:

$$\mathop{\textcircled{Σ}}\limits_{(X',Y')\in G_2} \mathop{\textcircled{Π}}\limits_{i=1}^{|X'|} d(x_i',y_i') = \mathop{\textcircled{Σ}}\limits_{(X',Y')\in G_2} \left[\mathop{\textcircled{Π}}\limits_{i=1}^{|X'|-1} d(x_i',y_i') \otimes d(\lambda,y_M)\right]$$

$$= \left[\mathop{\textcircled{Σ}}\limits_{(X',Y')\in G_2} \mathop{\textcircled{Π}}\limits_{i=1}^{|X'|-1} d(x_i',y_i')\right] \otimes d(\lambda,y_M)$$

$$= D(X,\underline{Y}) \otimes d(\lambda,y_M) \quad , \tag{A.5}$$

In every element in the set G_3, $p' = x_N$ and $q' = \lambda$. Arguing as in the previous cases, the third term in (A.2) becomes:

$$\mathop{\textcircled{Σ}}\limits_{(X',Y')\in G_3} \mathop{\textcircled{Π}}\limits_{i=1}^{|X'|} d(x_i',y_i') = \mathop{\textcircled{Σ}}\limits_{(X',Y')\in G_3} \left[\mathop{\textcircled{Π}}\limits_{i=1}^{|X'|-1} d(x_i',y_i') \otimes d(x_N,\lambda)\right]$$

$$= \left[\mathop{\textcircled{Σ}}\limits_{(X',Y')\in G_3} \mathop{\textcircled{Π}}\limits_{i=1}^{|X'|-1} d(x_i',y_i')\right] \otimes d(x_N,\lambda) \quad ,$$

$$= D(\underline{X},Y) \otimes d(x_N,\lambda) \tag{A.6}$$

Resubstituting (A.4 - A.6) in (A.2) proves the theorem.

CARLETON UNIVERSITY
School of Computer Science

Bibliography of SCS Reports

SCS-TR-1	THE DESIGN OF CP-6 PASCAL Jim des Rivieres and Wilf R. LaLonde, June 1982.
SCS-TR-2	SINGLE PRODUCTION ELIMINATION IN LR(1) PARSERS:A SYNTHESIS Wilf R. LaLonde, June 1982.
SCS-TR-3	A FLEXIBLE COMPILER STRUCTURE THAT ALLOWS DYNAMIC PHASE ORDERING Wilf R. LaLonde and Jim des Rivieres, June 1982.
SCS-TR-4	A PRACTICAL LONGEST COMMON SUBSEQUENCE ALGORITHM FOR TEXT COLLATION Jim des Rivieres, June 1982.
SCS-TR-5	A SCHOOL BUS ROUTING AND SCHEDULING PROBLEM Wolfgang Lindenberg, Frantisek Fiala, July 1982.
SCS-TR-6	ROUTING WITHOUT ROUTING TABLES Nicola Santoro, Ramez Khatib, July 1982.
SCS-TR-7	CONCURRENCY CONTROL IN LARGE COMPUTER NETWORKS Nicola Santoro, Hasan Hural, July 1982.
SCS-TR-8	ORDER STATISTICS ON DISTRIBUTED SETS Nicola Santoro, Jeffrey B. Sidney, July 1982.
SCS-TR-9	OLIGARCHICAL CONTROL OF DISTRIBUTED PROCESSING SYSTEMS Moshe Krieger, Nicola Santoro, August 1982.
SCS-TR-10	COMMUNICATION BANDS FOR SELECTION IN A DISTRIBUTED SET Nicola Santoro, Jeffrey B. Sidney, September 1982.
SCS-TR-11	A SIMPLE TECHNIQUE FOR CONVERTING FROM A PASCAL SHOP TO C SHOP Wilf R. LaLonde, John R. Pugh, November 1982.
SCS-TR-12	EFFICIENT ABSTRACT IMPLEMENTATIONS FOR RELATIONAL DATA STRUCTURES Nicola Santoro, December 1982.
SCS-TR-13	ON THE MESSAGE COMPLEXITY OF DISTRIBUTED PROBLEMS Nicola Santoro, December 1982.
SCS-TR-14	A COMMON BASIS FOR SIMILARITY MEASURES INVOLVING TWO STRINGS R.L. Kashyap and B.J. Oommen, January 1983.
SCS-TR-15	SIMILARITY MEASURES FOR SETS OF STRINGS R.L. Kashyap and B.J. Oommen, January 1983.