The Parameterized Complexity of Sequence Alignment and Consensus

Hans L. Bodlaender * Rodney G. Downey † Michael R. Fellows ‡
Harold T. Wareham §

Abstract

The LONGEST COMMON SUBSEQUENCE problem is examined from the point of view of parameterized computational complexity. There are several different ways in which parameters enter the problem, such as the number of sequences to be analyzed, the length of the common subsequence, and the size of the alphabet. Lower bounds on the complexity of this basic problem imply lower bounds on a number of other sequence alignment and consensus problems. At issue in the theory of parameterized complexity is whether a problem which takes input (x, k) can be solved in time $f(k) \cdot n^{\alpha}$ where α is independent of k (termed *fixed-parameter tractability*). It can be argued that this is the appropriate asymptotic model of feasible computability for problems for which a small range of parameter values covers important applications — a situation which certainly holds for many problems in biological sequence analysis. Our main results show that: (1) The LONGEST COMMON SUBSEQUENCE (LCS) parameterized by the number of sequences to be analyzed is hard for $W[t]$ for all t. (2) The LCS problem problem, parameterized by the length of the common subsequence, belongs to $W[P]$ and is hard for $W[2]$. (3) The LCS problem parameterized both by the number of sequences and the length of the common subsequence, is complete for $W[1]$. All of the above results are obtained for unrestricted alphabet sizes. For alphabets of a fixed size, problems (2) and (3) are fixed-parameter tractable. We conjecture that (1) remains hard.

---

*Computer Science Department, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. Research partially supported by the ESPRIT Basic Research Actions of the EC under contract 7141 (project ALCOM II). hansb@cs.ruu.nl

†Mathematics Department, Victoria University, P.O. Box 600, Wellington, New Zealand. Research supported in part by a grant from Victoria University IGC, by the United States / New Zealand Cooperative Science Foundation under grant INT 90-20558. downey@math.vuw.ac.nz

‡Computer Science Department, University of Victoria, Victoria, British Columbia V8W 3P6, Canada. Research supported in part by the National Science and Engineering Council of Canada and by the United States National Science Foundation under grant MIP-8919312. mfellows@csr.uvic.ca

§Computer Science Department, University of Victoria, Victoria, British Columbia V8W 3P6, Canada. harold@sanjuan.uvic.ca

1 Introduction

The computational problem of finding the longest common subsequence of a set of k strings (the LCS problem) has been studied extensively over the last twenty years (see [Hir83,IF92] and references). This problem has many applications. When $k = 2$, the longest common subsequence is a measure of the similarity of two strings and is thus useful in in molecular biology, pattern recognition, and text compression [San72,LF78,Mai78]. The version of LCS in which the number of strings is unrestricted is also useful in text compression [Mai78], and is a special case of the multiple sequence alignment and consensus subsequence discovery problems in molecular biology [Pev92,DM93a,DM93b].

To date, most research has focused on deriving efficient algorithms for the LCS problem when $k = 2$ (see [Hir83,IF92] and references). Most of these algorithms are based on the dynamic programming approach [PM92], and require quadratic time. Though the k-unrestricted LCS problem is NP-complete [Mai78], certain of the algorithms for the $k = 2$ case have been extended to yield algorithms that require $O(n^{(k-1)})$ time and space, where n is the length of the longest of the k strings (see [IF92] and references; see also [Bae91]).

In this paper, we analyze the *Longest common subsequence* problem from the point of view of parameterized complexity theory introduced in [DF92]. The parameterizations of the *Longest Common Subsequence* problem that we consider are defined as follows.

LONGEST COMMON SUBSEQUENCE (LCS-1, LCS-2 and LCS-3)
Input: A set of k strings X_1, ..., X_k over an alphabet Σ, and a positive integer m.
Parameter 1: k (We refer to this problem as LCS-1.)
Parameter 2: m (We refer to this problem as LCS-2.)
Parameter 3: (k, m) (We refer to this problem as LCS-3.)
Question: Is there a string $X \in \Sigma^*$ of length at least m that is a subsequence of X_i for $i = 1, ..., k$?

Our results are summarized in Table 1.

In §2 we give some background on parameterized complexity theory. In §3 we detail the proof that LCS-3 is complete for W[1]. This implies that LCS-1 and LCS-2 are W[1]-hard, results which can be improved by further arguments to show that LCS-1 is hard for $W[t]$ for all t, and that LCS-2 is hard for $W[2]$. Concretely, this means none of these three parameterized versions of LCS is fixed-parameter tractable unless many other well-known and apparently resistant problems are also fixed-parameter tractable.

Table 1: The Fixed-Parameter Complexity of the LCS Problem

Problem	Parameter	Alphabet Size $\|\Sigma\|$	
		Unbounded	Fixed
LCS-1	k	W[t]-hard, $t \geq 1$	?
LCS-2	m	W[2]-hard	FPT
LCS-3	k, m	W[1]-complete	FPT

2 Parameterized Computational Complexity

The theory of parameterized compuational complexity is motivated by the observation that many NP-complete problems take as input two objects, for example, perhaps a graph G and and integer k. In some cases, e.g., Vertex cover, the problem can be solved in linear time for every fixed parameter value, and is well-solved for problems with $k \leq 20$. For other problems, for example Clique and Minimum dominating set we have the contrasting situation where the best known algorithms are based on brute force, essentially, and require time $\Omega(n^k)$. If $P = NP$ then all three of these problems are fixed-parameter tractable. The theory of parameterized computational complexity explores the apparent qualitative difference between these problems (for fixed parameter values). It is particularly relevant to problems where a small range of parameter values cover important applications — this is certainly the case for many problems in computational biology. For these the theory offers a more sensitive view of tractability vs. apparent intractability than the theory of NP-completeness.

2.1 Parameterized Problems and Fixed-Parameter Tractability

A *parameterized problem* is a set $L \subseteq \Sigma^* \times \Sigma^*$ where Σ is a fixed alphabet. For convenience, we consider that a parameterized problem L is a subset of $L \subseteq \Sigma^* \times N$. For a parameterized problem L and $k \in N$ we write L_k to denote the associated fixed-parameter problem $L_k = \{x|(x,k) \in L\}$.

Definition 1 *We say that a parameterized problem L is (uniformly) fixed-parameter tractable if there is a constant α and an algorithm Φ such that Φ decides if $(x,k) \in L$ in time $f(k)|x|^\alpha$ where $f : N \rightarrow N$ is an arbitrary function.*

2.2 Problem Reductions

A direct proof that a problem such as MINIMUM DOMINATING SET is not fixed-parameter tractable would imply $P \neq NP$. Thus a completeness program is reasonable.

Definition 2 *Let A, B be parameterized problems. We say that A is (uniformly many:1) reducible to B if there is an algorithm Φ which transforms (x, k) into $(x', g(k))$ in time $f(k)|x|^\alpha$, where $f, g : N \to N$ are arbitrary functions and α is a constant independent of k, so that $(x, k) \in A$ if and only if $(x', g(k)) \in B$.*

It is easy to see that if A reduces to B and B is fixed parameter tractable then so too is A. It is important to note that there are two ways in which parameterized reductions differ from familiar P-time reductions: (1) the reduction may be polynomial in n, but (for example) exponential in the parameter k, and (2) the slice A_k must be mapped to a single slice $B_{g(k)}$ (unlike NP-completeness reductions which may map k to $k' = n - k$, for example).

2.3 Complexity Classes

The classes are intuitively based on the complexity of the circuits required to check a solution, or alternatively, the "natural logical depth" of the problem.

Definition 3 *A Boolean circuit is of mixed type if it consists of circuits having gates of the following kinds.*
(1) Small gates: not gates, and gates and or gates with bounded fan-in. We will usually assume that the bound on fan-in is 2 for and gates and or gates, and 1 for not gates.
(2) Large gates: and gates and or gates with unrestricted fan-in.

Definition 4 *The depth of a circuit C is defined to be the maximum number of gates (small or large) on an input-output path in C. The weft of a circuit C is the maximum number of large gates on an input-output path in C.*

Definition 5 *We say that a family of decision circuits F has bounded depth if there is a constant h such that every circuit in the family F has depth at most h. We say that F has bounded weft if there is constant t such that every circuit in the family F has weft at most t. The weight of a boolean vector x is the number of 1's in the vector.*

Definition 6 *Let F be a family of decision circuits. We allow that F may have many different circuits with a given number of inputs. To F we associate the parameterized circuit problem $L_F = \{(C,k) : C$ accepts an input vector of weight $k\}$.*

Definition 7 *A parameterized problem L belongs to $W[t]$ if L reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t,h)$ of mixed type decision circuits of weft at most t, and depth at most h, for some constant h.*

Definition 8 *A parameterized problem L belongs to $W[P]$ if L reduces to the circuit problem L_F, where F is the set of all circuits (no restrictions).*

We designate the class of fixed-parameter tractable problems FPT.

The framework above describes a hierarchy of parameterized complexity classes

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$$

for which there are many natural hard or complete problems [DF92].

For example, all of the following problems are now known to be complete for $W[1]$: Square tiling, Independent set, Clique, and Bounded post correspondence problem, k-Step derivation for context-sensitive grammars, Vapnik-Chervonenkis dimension, and the k-Step halting problem for nondeterministic Turing machines [CCDF93,DEF93,DFKHW93]. Thus, any one of these problems is fixed-parameter tractable if and only if all of the others are; and none of the problems for which we here prove W hardness results are fixed-parameter tractable unless all of these are also. Dominating set is complete for $W[2]$ [DF92]. Fixed parameter tractability for Dominating set, or any other $W[2]$-hard problem implies fixed parameter tractability for all problems in $W[1]$ mentioned above, and all other problems in $W[2] \supseteq W[1]$.

3 The Reductions

In some sense the most basic of the three parameterized versions of LCS that we consider is LCS-3, since hardness results for this problem immediately imply hardness results for LCS-1 and LCS-2.

Theorem 1 *LCS-3 is complete for $W[1]$.*

Proof. Membership in $W[1]$ can be seen by a reduction to WEIGHTED CNF SATISFIABILITY for expressions having bounded clause size. By padding with new symbols or by repeating some of the X_i, we can assume for convenience (with polynomially bounded blow-up) that $k = m$. The idea is to use a truth assignment of weight k^2 to indicate the k positions in each of the k strings of an instance of LCS-3 that yield a common subsequence of length k.

The details are as follows. Let $X_1, \ldots, X_k$ be an instance of LCS-3. By a trivial padding with symbols having only a single occurence we may assume that the strings X_i are all of length n. Let $a[i,j]$ denote the j^{th} symbol of X_i. Let $B = \{b[i,j,r] : 1 \leq i \leq k,\ 1 \leq j \leq n,\ 1 \leq r \leq k\}$ be a set of boolean variables. The interpretation we intend for the variable $b[i,j,r]$ is that the r^{th} symbol $x[r]$ of a length k common subsequence $X = x[1] \cdots x[k]$ occurs as the symbol $a[i,j]$ in the string X_i, that is, $x[j] = a[i,j]$. Let B_i be the set of elements $b[i,j,r]$ with first index i.

Let $E = E_1 E_2 E_3$ be the boolean expression over the set of variables B where

$$E_1 = \prod_{i=1}^{k} \prod_{r=1}^{k} \prod_{1 \leq j < j' \leq n} (\neg b[i,j,r] + \neg b[i,j',r])$$

$$E_2 = \prod_{1=1}^{k} \prod_{j=1}^{n} \prod_{1 \leq r < r' \leq k} (\neg b[i,j,r] + \neg b[i,j,r'])$$

$$E_3 = \prod_{r=1}^{k} \prod_{1 \leq i < i' \leq k} \prod_{1 \leq j \leq j' \leq n} \prod_{a[i,j] \neq a[i',j']} (\neg b[i,j,r] + \neg b[i',j',r])$$

We claim that E has a weight k^2 truth assignment if and only if the X_i have a common subsequence of length k. It is easy to verify that a truth assignment corresponding to a length k common subsequence according to our intended interpretation of the boolean variables satisfies E. For the converse direction, suppose τ is a weight k^2 truth assignment that satisfies E. The clauses of E_1 insure (by the Pigeonhole Principle) that no more than k variables of B_i are set *true* for $i = 1, ..., k$. Consequently there must be exactly k variables set to *true* in each B_i, and since E_2 is satisfied, these must indicate k distinct positions in X_i according to our interpretation. The clauses of E_3 insure that the corresponding subsequence symbols in the k strings are the same.

To show $W[1]$-hardness we reduce from CLIQUE. Let $G = (V, E)$ be a graph for which we wish to determine whether G has a k-clique. We show how to construct a family $\mathcal{F}_G$ of $k' = f(k)$ sequences over an alphabet Σ that have a common subsequence of length $k'' = g(k)$ if and only G contains a k-clique. Assume for convenience that the vertex set of G is $V = \{1, \ldots, n\}$.

The Alphabet We first describe the alphabet $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$. We refer to these as *vertex symbols* (Σ_1), *edge symbols* (Σ_2), *vertex position symbols* (Σ_3), and *edge position symbols* (Σ_4).

$$\Sigma_1 = \{\alpha[p,q,r] :\ 1 \le p \le k,\ 0 \le q \le 1,\ 1 \le r \le n\}$$

$$\Sigma_2 = \{\beta[i,j,q,u,v] :\ 1 \le i < j \le k,\ 0 \le q \le 1,\ 1 \le u < v \le n,\ uv \in E\}$$

$$\Sigma_3 = \{\gamma[p,q,b] :\ 1 \le p \le k,\ 0 \le q \le 1, 0 \le b \le 1\}$$

$$\Sigma_4 = \{\delta[i,j,q,b] :\ 1 \le i < j \le k,\ 0 \le q \le 1,\ 0 \le b \le 1\}$$

We will use the following shorthand notation to refer to various subsets of Σ. The notation indicates which indices are held fixed to some value, with "*" indicating that the index should vary over its range of definition in building the set. For example, $\Sigma_1[p,*,r] = \{\alpha[p,q,r] :\ 0 \le q \le 1\}$ is the set of two elements with the first and third indices fixed at p and r, respectively.

An Example of a Clique Representation The sequences in $\mathcal{F}$ are constructed in such a way that the k-cliques in G (considered with vertices in ascending order) are in 1:1 correspondence with the common subsequences of length k''. It will be useful in motivating the construction to consider an example of this intended correspondence. Consider a graph having a 3-clique on the vertices $\{a,b,c\}$.

This 3-clique would be represented by the following common subsequence $\sigma(a,b,c)$, which we describe according to a hierarchy of factorizations. (Exponential notation indicates repetition of a symbol.)

$$\sigma(a,b,c) = \langle\text{first vertex}\rangle\langle\text{second vertex}\rangle\langle\text{third vertex}\rangle$$

where

$$\langle\text{first vertex}\rangle = \langle\text{vertex 1}\rangle\langle\text{edge (1,2)}\rangle\langle\text{edge (1,3)}\rangle\langle\text{vertex 1 echo}\rangle$$

$$\langle\text{second vertex}\rangle = \langle\text{vertex 2}\rangle\langle\text{edge (1,2) echo}\rangle\langle\text{edge (2,3)}\rangle\langle\text{vertex 2 echo}\rangle$$

$$\langle\text{third vertex}\rangle = \langle\text{vertex 3}\rangle\langle\text{edge (1,3) echo}\rangle\langle\text{edge (2,3) echo}\rangle\langle\text{vertex 3 echo}\rangle$$

and where the constituent subsequences over Σ are

$$\begin{aligned}
\langle\text{vertex 1}\rangle &= \gamma[1,0,0]^w\alpha[1,0,a]\gamma[1,0,1]^w\\
\langle\text{edge (1,2)}\rangle &= \delta[1,2,0,0]^w\beta[1,2,0,a,b]\delta[1,2,0,1]^w\\
\langle\text{edge (1,3)}\rangle &= \delta[1,3,0,0]^w\beta[1,3,0,a,c]\delta[1,3,0,1]^w\\
\langle\text{vertex 1 echo}\rangle &= \gamma[1,1,0]^w\alpha[1,1,a]\gamma[1,1,1]^w\\
\langle\text{vertex 2}\rangle &= \gamma[2,0,0]^w\alpha[2,0,b]\gamma[2,0,1]^w\\
\langle\text{edge (1,2) echo}\rangle &= \delta[1,2,1,0]^w\beta[1,2,1,a,b]\delta[1,2,1,1]^w\\
\langle\text{edge (2,3)}\rangle &= \delta[2,3,0,0]^w\beta[2,3,0,b,c]\delta[2,3,0,1]^w
\end{aligned}$$

$$
\begin{aligned}
\langle \text{vertex 2 echo} \rangle &= \gamma[2,1,0]^w \alpha[2,1,b] \gamma[2,1,1]^w \\
\langle \text{vertex 3} \rangle &= \gamma[3,0,0]^w \alpha[3,0,c] \gamma[3,0,1]^w \\
\langle \text{edge (1,3) echo} \rangle &= \delta[1,3,1,0]^w \beta[1,3,1,a,c] \delta[1,3,1,1]^w \\
\langle \text{edge (2,3) echo} \rangle &= \delta[2,3,1,0]^w \beta[2,3,1,b,c] \delta[2,3,1,1]^w \\
\langle \text{vertex 3 echo} \rangle &= \gamma[3,1,0]^w \alpha[3,1,c] \gamma[3,1,1]^w
\end{aligned}
$$

In the above, the position symbols are repeated $w = w(k)$ times for reasons useful for the correctness argument concerning the reduction.

The Target Parameters There are $f_1(k) = 2k + k(k-1) = k^2 + k$ matched pairs of position symbols (in Σ_3 and Σ_4). We take $w = f_1(k)^2 + 1$, $k' = f_1(k) + 2$, and $k'' = (2w+1)f_1(k)$.

Symbol Subsets and Operations It is convenient to introduce a linear ordering on Σ that corresponds to the "natural" order in which the various symbols occur, as illustrated by the example above. We can achieve this by defining a "weight" on the symbols of Σ and then ordering the symbols by weight.

Let $N = 2kn$ (a value conveniently larger than k and n). Define the *weight* $||a||$ of a symbol $a \in \Sigma$ by

$$
||a|| = \begin{cases}
pN^6 + qN^5 + r & \text{if } a = \alpha[p,q,r] \in \Sigma_1 \\
q'iN^6 + qjN^6 + q'N^4 + q'jN^3 + qiN^3 + uN + v & \text{if } a = \beta[i,j,q,u,v] \in \Sigma_2 \\
pN^6 + qN^5 + bN^2 & \text{if } a = \gamma[p,q,b] \in \Sigma_3 \\
q'iN^6 + qjN^6 + q'N^4 + q'jN^3 + qiN^3 + bN^2 & \text{if } a = \delta[i,j,q,b] \in \Sigma_4
\end{cases}
$$

where $q' = (q-1)^2$.

Define a linear order on Σ by $a < b$ if and only if $||a|| < ||b||$. The reader can verify that, assuming $a < b < c$, the symbols of the example sequence $\sigma(a,b,c)$ described above occur in ascending order.

For $a, b \in \Sigma$, $a < b$, we define the *segment* $\Sigma(a,b)$ to be $\Sigma(a,b) = \{e \in \Sigma : a \leq e \leq b\}$, and we define similarly the segments $\Sigma_i(a,b)$.

If , is a set of symbols, then $\langle *, * \rangle$ denotes an arbitrary string which contains as a subsequence every string of length m over , (such as a string which simply runs through , m times in any order).

If , $\subseteq \Sigma$, let $(\uparrow$,) be the string of length |, | which consists of one occurence of each symbol in , in ascending order, and let $(\downarrow$,) be the string of length |, | which consists of one occurence of each symbol in , in descending order.

String Gadgets We next describe some "high level" component subsequences for the construction. In the following let $\updownarrow$ denote either $\uparrow$ or $\downarrow$. Product notation is interpreted as refering to concatenation. In describing some of the components we will use $\uparrow$ lex to denote increasing lexicographic order and $\downarrow$ lex to denote decreasing lexicographic order.

Vertex and Edge Selection Gadgets

$$\langle\updownarrow \text{ vertex } p\rangle = \gamma[p,0,0]^w(\updownarrow \Sigma_1[p,0,*])\gamma[p,0,1]^w$$

$$\langle\updownarrow \text{ vertex } p \text{ echo}\rangle = \gamma[p,1,0]^w(\updownarrow \Sigma_1[p,1,*])\gamma[p,1,1]^w$$
$$\langle\updownarrow \text{ edge } (i,j)\rangle = \delta[i,j,0,0]^w(\updownarrow \Sigma_2[i,j,0,*,*])\delta[i,j,0,1]^w$$
$$\langle\updownarrow \text{ edge } (i,j) \text{ echo}\rangle = \delta[i,j,1,0]^w(\updownarrow \Sigma_2[i,j,1,*,*])\delta[i,j,1,1]^w$$
$$\langle\updownarrow \text{ edge } (i,j) \text{ from } u\rangle = \delta[i,j,0,0]^w(\updownarrow \Sigma_2[i,j,0,u,*])\delta[i,j,0,1]^w$$
$$\langle\updownarrow \text{ edge } (i,j) \text{ to } v\rangle = \delta[i,j,1,0]^w(\updownarrow \Sigma_2[i,j,1,*,v])\delta[i,j,1,1]^w$$

Control and Selection Assemblies

$$\begin{aligned}\langle\updownarrow \text{ control } p\rangle &= \langle\updownarrow \text{ vertex } p\rangle\left(\prod_{s=1}^{p-1}\langle\updownarrow \text{ edge } (s,p) \text{ echo}\rangle\right)\\ &\cdot \left(\prod_{s=p+1}^{k}\langle\updownarrow \text{ edge } (p,s)\rangle\right)\langle\updownarrow \text{ vertex } p \text{ echo}\rangle\end{aligned}$$

$$\begin{aligned}\langle\uparrow \text{ choice } p\rangle &= \prod_{x=1}^{n}\Bigg(\gamma[p,0,0]^w\alpha[p,0,x]\gamma[p,0,1]^w\prod_{t=1}^{p-1}\langle\uparrow \text{ edge } (t,p) \text{ to } x\rangle\\ &\cdot \prod_{t=p+1}^{k}\langle\uparrow \text{ edge } (p,t) \text{ from } x\rangle\gamma[p,1,0]^w\alpha[p,1,x]\gamma[p,1,1]^w\Bigg)\end{aligned}$$

$$\begin{aligned}\langle\downarrow \text{ choice } p\rangle &= \prod_{x=n}^{\text{down to } 1}\Bigg(\gamma[p,0,0]^w\alpha[p,0,x]\gamma[p,0,1]^w\prod_{t=1}^{p-1}\langle\downarrow \text{ edge } (t,p) \text{ to } x\rangle\\ &\cdot \prod_{t=p+1}^{k}\langle\downarrow \text{ edge } (p,t) \text{ from } x\rangle\gamma[p,1,0]^w\alpha[p,1,x]\gamma[p,1,1]^w\Bigg)\end{aligned}$$

Edge Symbol Pairing Gadget

$$\langle\text{edge } (i,j) \text{ from } u \text{ to } v\rangle = \beta[i,j,0,u,v]\,(*\Sigma(\delta[i,j,0,1],\delta[i,j,1,0])*)\,\beta[i,j,1,u,v]$$

The Reduction We may now describe the reduction. The instance of LCS-3 consists of strings which we may consider as belonging to three subsets: *Control*, *Selection* and *Check*. The two strings in the *Control* set are

$$X_1 = \prod_{t=1}^{k} \langle \uparrow \text{control } t \rangle$$

$$X_2 = \prod_{t=1}^{k} \langle \downarrow \text{control } t \rangle$$

The $2k$ strings in the *Selection* set are, for $p = 1, ..., k$

$$Y_p = \left(\prod_{t=1}^{p-1} \langle \uparrow \text{control } t \rangle \right) \langle \uparrow \text{choice } p \rangle \left(\prod_{t=p+1}^{k} \langle \uparrow \text{control } t \rangle \right)$$

$$Y'_p = \left(\prod_{t=1}^{p-1} \langle \downarrow \text{control } t \rangle \right) \langle \downarrow \text{choice } p \rangle \left(\prod_{t=p+1}^{k} \langle \uparrow \text{control } t \rangle \right)$$

The $2\binom{k}{2} = k(k-1)$ strings in the *Check* set are, for $1 \le i < j \le k$

$$\begin{aligned} Z_{i,j} \quad = \quad & \left(\prod_{t=1}^{i-1} \langle \uparrow \text{control } t \rangle \right) \langle \uparrow \text{vertex } i \rangle \left(\prod_{s=1}^{i-1} \langle \uparrow \text{edge } (s,i) \text{ echo} \rangle \right) \left(\prod_{s=i+1}^{j-1} \langle \uparrow \text{edge } (i,s) \rangle \right) \\ \cdot \quad & \delta[i,j,0,0]^w \prod_{\substack{1 \le u < v \le n \\ uv \in E}}^{\text{lex}\uparrow} \langle \text{edge } (i,j) \text{ from } u \text{ to } v \rangle \\ \cdot \quad & \delta[i,j,1,1]^w \left(\prod_{s=i+1}^{j-1} \langle \uparrow \text{edge } (s,j) \text{ echo} \rangle \right) \left(\prod_{s=j+1}^{k} \langle \uparrow \text{edge } (j,s) \rangle \right) \\ \cdot \quad & \langle \uparrow \text{vertex } j \text{ echo} \rangle \prod_{t=j+1}^{k} \langle \uparrow \text{control } t \rangle \end{aligned}$$

$$\begin{aligned} Z'_{i,j} \quad = \quad & \left(\prod_{t=1}^{i-1} \langle \downarrow \text{control } t \rangle \right) \langle \downarrow \text{vertex } i \rangle \left(\prod_{s=1}^{i-1} \langle \downarrow \text{edge } (s,i) \text{ echo} \rangle \right) \left(\prod_{s=i+1}^{j-1} \langle \downarrow \text{edge } (i,s) \rangle \right) \\ \cdot \quad & \delta[i,j,0,0]^w \prod_{\substack{1 \le u < v \le n \\ uv \in E}}^{\text{lex}\downarrow} \langle \text{edge } (i,j) \text{ from } u \text{ to } v \rangle \\ \cdot \quad & \delta[i,j,1,1]^w \left(\prod_{s=i+1}^{j-1} \langle \downarrow \text{edge } (s,j) \text{ echo} \rangle \right) \left(\prod_{s=j+1}^{k} \langle \downarrow \text{edge } (j,s) \rangle \right) \\ \cdot \quad & \langle \downarrow \text{vertex } j \text{ echo} \rangle \prod_{t=j+1}^{k} \langle \downarrow \text{control } t \rangle \end{aligned}$$

We comment that the key difference between $Z_{i,j}$ and $Z'_{i,j}$ is that in $Z_{i,j}$ the edge symbol pairing gadgets occur in increasing lexicographic order, and in $Z'_{i,j}$ the gadgets are in decreasing lexicographic order.

Proof of Correctness Where S_1 and S_2 are strings of symbols, let $l(S_1, S_2)$ denote the maximum length of a common subsequence of S_1 and S_2.

In the Control Strings X_1 and X_2 we distinguish certain substrings that we term *positions*. Note that both of these strings are formed as the concatenation of four different kinds of substrings: ⟨vertex⟩, ⟨vertex echo⟩, ⟨edge⟩ and ⟨edge echo⟩, and that each of these "vertex and edge selection" substrings begins and ends with a matched pair of substrings of repeated symbols from Σ_3 (in the case of vertex selection), or from Σ_4 (in the case of edge selection). These matched pairs of position symbol substrings determine a *position* — note that these position symbol substrings (and therefore the positions defined) occur in the same order in X_1 and X_2. Thus there are $k(2 + k - 1) = k^2 + k$ positions.

Between a matched pair of position symbol substrings in X_1 there is a set of symbols in increasing order that we will term a *set of (vertex or edge) stairs*, and in X_2 in the corresponding position there occurs the same set of symbols in decreasing order. The proof of the following claim is trivial.

Claim 1. Suppose Σ is a linearly ordered finite alphabet, and that $S\uparrow$ is the string consisting of the symbols of Σ in increasing order, and that $S\downarrow$ is the symbols of Σ in decreasing order. Then $l(S\uparrow, S\downarrow) = 1$. □

Claim 2. A common subsequence C of the control sequences X_1 and X_2 of maximum length l satisfies the conditions: (1) $l = k''$, and (2) C consists of the position symbol substrings (common to X_1 and X_2) together with one symbol in each position defined by these substrings.

Proof. It is clear that $l \geq k''$ because there are many different common subsequences of length k'' consisting of all the position symbol substrings (which are the same in X_1 and X_2) together with a single choice of vertex or edge symbol in each position. Now suppose there is a common subsequence C of length greater than k'' and fix attention on subsequences C_1 of X_1 and C_2 of X_2 that are isomorphic to C (for the reason that C might occur in more than one way as a subsequence). Then C_1 must contain two vertex or edge symbols ϵ_1 and ϵ_2 that occur on the same set of stairs in X_1. By Claim 1, these two symbols, considered now in C_2, cannot occur on the same set of stairs in X_2. This implies that any position symbols between ϵ_1 and ϵ_2 in X_2 do not belong to C_2. Consequently, there are at least $2w$ position symbols of X_2 that do not occur in $C = C_2$. But in order for the length of C to be at least k'', this means that C must contain more than $f_1(k)^2$ vertex and edge symbols. By the Pigeonhole Principle, there must therefore be a set of stairs in X_1 that contains $m > f_1(k)$ vertex or edge symbols of C_1. By Claim 1, no more than one of the corresponding symbols in C_2 can occur on any set of stairs in X_2, and therefore X_2 must have at least m sets of stairs, a contradiction. This establishes (1), and furthermore shows that no two symbols of a common subsequence of length k'' can occur on the same set of stairs. Thus (2) may also be concluded by observing that there must be at least one vertex or edge symbol from each set of stairs, else the length of C would be less than k''. □

By Claim 2, if C is a common subsequence of X_1 and X_2 of length k'', we may refer unambiguously to the vertices and edges represented in the various positions of C. In particular, note that these positions occur in k *vertex units*, each of which consists of an *initial vertex position*, followed by $k-1$ *edge* and *edge echo positions* and concluding with a *terminal vertex echo position*. If uv is an edge of the graph with $u < v$, then we refer to u as the *initial vertex* and to v as the *terminal vertex* of the edge.

Claim 3. If C is a subsequence of length k'' common to the Control and Selection sets, then in each vertex unit: (1) the vertex u represented in the initial vertex position is also represented in the terminal vertex echo position, (2) each edge represented in an edge echo position has terminal vertex u, and (3) each edge represented in an edge position has initial vertex u.

Proof. Suppose C is a subsequence of length k'' common to X_1 and X_2. We argue that if C is also common to Y_p and Y'_p then the statements of the Lemma are satisfied for the p^{th} vertex unit. Let C_p and C'_p denote specific subsequences of Y_p and Y'_p, respectively, with $C = C_p = C'_p$.

The strings Y_p and Y'_p differ from the Control strings X_1 and X_2, respectively, only in the replacement of a $\langle \updownarrow \text{control } p \rangle$ gadget with a $\langle \updownarrow \text{choice } p \rangle$ gadget. In particular, the position symbols in the other constituent substrings occur in the same way in all four strings, and so by Claim 2, C_p (C'_p) must include all of the length w position symbol substrings in Y_p (Y'_p) occuring outside of $\langle \uparrow \text{choice } p \rangle$ ($\langle \downarrow \text{choice } p \rangle$). Furthermore, C_p must contain precisely two vertex symbols α and α', appropriately positioned, from $\langle \uparrow \text{choice } p \rangle$, and C'_p must contain the same two (and no other) vertex symbols from $\langle \downarrow \text{choice } p \rangle$.

The subsequence of Y_p consisting of all the vertex symbols in $\langle \uparrow \text{choice } p \rangle$ is the vertex index increasing sequence

$$S = \prod_{x=1}^{n} (\alpha[p, 0, x]\alpha[p, 1, x])$$

and the subsequence of Y'_p consisting of all the vertex symbols in $\langle \downarrow \text{choice } p \rangle$ is the vertex index decreasing sequence

$$S' = \prod_{x=n}^{1} (\alpha[p, 0, x]\alpha[p, 1, x])$$

The only possibility for α and α' to be common to S and S' is for α and α' to represent the same vertex u, that is, $\alpha = \alpha[p, 0, u]$ and $\alpha' = \alpha[p, 1, u]$. This establishes (1).

Consider the position symbols occuring in Y_p between α and α' in C_p, and occuring in Y'_p between α and α' in C'_p. Since these must occur in C (by Lemma 2) and this can happen in only one way, all of these position symbols must belong to C_p and C'_p, respectively. This insures (2) and (3). □

The length w substrings of the position symbols $\delta[i,j,0,0]$ and $\delta[i,j,0,1]$ in C define the $(i,j)^{th}$ edge position in the i^{th} vertex unit and the length w substrings of the position symbols $\delta[i,j,1,0]$ and $\delta[i,j,1,1]$ in C define the $(i,j)^{th}$ edge echo position in the j^{th} vertex unit. We term these a *corresponding pair* of edge and edge echo positions.

Claim 4. If C is a subsequence of length k'' common to the Control, Selection and Check sets, then for each corresponding pair of an edge position and an edge echo position, the same edge must be represented in the two positions.

Proof. Suppose C is a subsequence of length k'' common to the Control and Selection sets. We argue that if C is also common to $Z_{i,j}$ and $Z'_{i,j}$ then Lemma holds for the $(i,j)^{th}$ corresponding pair of positions. Let $C_{i,j}$ and $C'_{i,j}$ denote specific subsequences of $Z_{i,j}$ and $Z'_{i,j}$ isomorphic to C.

It is convenient to consider $Z_{i,j}$ (and similarly $Z'_{i,j}$) under the factorization $Z_{i,j} = Z_{i,j}(1)Z_{i,j}(2)Z_{i,j}(3)$ where

$$Z_{i,j}(2) = \prod_{\substack{1 \le u < v \le n \\ uv \in E}}^{\text{lex}\uparrow} \langle \text{edge } (i,j) \text{ from } u \text{ to } v \rangle$$

and where $Z_{i,j}(1)$ and $Z_{i,j}(3)$ are the appropriately defined prefix and suffix (respectively) of $Z_{i,j}$.

Since none of the position symbols in $Z_{i,j}(1)$ or $Z_{i,j}(3)$ occur in $Z_{i,j}(2)$, all of the position symbols in $Z_{i,j}(1)$ and $Z_{i,j}(3)$ must belong to $C_{i,j}$. Similarly, all of the position symbols in $Z'_{i,j}(1)$ and $Z'_{i,j}(3)$ must belong to $C'_{i,j}$. This implies, by Lemma 2, that $C_{i,j} \cup Z_{i,j}(2) = C'_{i,j} \cup Z'_{i,j}(2)$ begins with a symbol $\beta[i,j,0,u,v]$ and ends with a symbol $\beta[i,j,0,x,y]$. We argue that necessarily $u = x$ and $v = y$.

From the fact that $\beta[i,j,1,x,y]$ follows $\beta[i,j,0,u,v]$ in $Z_{i,j}(2)$, and from the construction of the latter in increasing lexicographic order, we may deduce that (u,v) precedes (x,y) lexicographically. Similarly, since $Z'_{i,j}(2)$ is constructed in decreasing lexicographic order, we obtain that (x,y) precedes (u,v), and therefore $(x,y) = (u,v)$. □

We now argue the correctness of the reduction as follows. If G has a k-clique, then it is easily seen that there is a common subsequence of length k'' in which the k vertex units represent the vertices of the clique, and the edge and edge echo positions within each vertex unit represent the edges incident on the represented vertex of the unit in increasing lexicographic order. (Each edge is thus represented twice, in the vertex units corresponding to its endpoints, first in an edge position in the initial vertex unit, and second in an edge echo position in the terminal vertex unit.)

Conversely, suppose there is a common subsequence C of length k''. By Claims 2 and 3, C represents a sequence of k vertices of G. That these must be a clique in G follows from Claim 4 and the definition of the "edge from" and "edge to" gadgets, which restrict the edges represented in a vertex unit to those present in the graph and for which the vertex is, respectively, initial or terminal. That completes the proof. $\square$

Theorem 1 implies immediately that LCS-1 and LCS-2 are hard for $W[1]$, but it is possible to say more about the parameterized complexity of these problems. Our theorem for LCS-1, interestingly, provides the starting point for a number of other hardness reductions in parameterized complexity theory, such as the results that TRIANGULATING COLORED GRAPHS, INTERVALIZING COLORED GRAPHS and BANDWIDTH are hard for $W[t]$ for all t [BFH94].

Theorem 2 *LCS-1 is hard for $W[t]$ for all t.*

Proof. By the results of [DF92] we may take the source instance of the reduction to be a t-normalized expression E and a positive integer k, where t is even and E is monotone. Let n denote the number of variables of E. By simple padding we may assume that E has the form:

$$E = \wedge_{i_1=1}^{n} \vee_{i_2=1}^{n} \cdots \wedge_{i_{t-1}=1}^{n} \vee_{i_t=1}^{n} l[i_1, \ldots, i_t]$$

Let $V = \{u_1, ..., u_n\}$ denote the set of variables of E. Thus in the above expression for E, $l[i_1, \ldots, i_t]$ is always a positive literal, that is, an element of V. We show how to produce an instance of LCS-1 consisting of $\binom{k}{2} + 2k + 2$ strings that have a common subsequence of length m if and only if E has a weight k truth assignment, with m described:

$$m = 3k + 3n^{t/2} + 2\sum_{j=0}^{t} c(j)w(j,t)$$

where

$$c(j) = n^{\lceil j/2 \rceil}$$

and

$$w(j,t) = n^{2t(t-j)}$$

We will use the following notation for indexing. The set $\{1, ..., n\}$ is denoted as $[n]$. By $[n]^r$ we mean the set $\{\alpha = (a_1, ..., a_r) :\ 1 \le a_i \le n \text{ for } 1 \le i \le r\}$. By $[n]^0$ we denote the singleton set $\{\epsilon\}$ where ϵ denotes the unique vector of length 0 over $[n]$. If $\alpha \in [n]^s$ and $b \in [n]$ with $\alpha = (a_1, ..., a_s)$, then we write $\alpha.b$ to denote $(a_1, ..., a_s, b) \in [n]^{s+1}$. We consider that $[n]^r$ is ordered lexicographically. As in the proof of Theorem 1, we will use $\uparrow$ lex to denote increasing lexicographic order and $\downarrow$ lex to denote decreasing lexicographic order. We make use of the index set I defined

$$I = \bigcup_{r=1}^{t} [n]^r$$

We say that $\alpha \in I$ is *even* if $\alpha \in [n]^r$ for r even, otherwise α is termed *odd*. If $\alpha \in [n]^r$ then we write $|\alpha| = r$ and term this the *rank* of α. If $\alpha, \beta \in I$ and α is a proper prefix of β then we write $\alpha \prec \beta$.

The Alphabet

The alphabet Σ for target instance of LCS-1 can be expressed as the union

$$\begin{aligned}
\Sigma &= \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4 \cup \Sigma_5 \\
\text{where} & \\
\Sigma_1 &= \{c[j], c'[j] : \ 1 \leq j \leq k\} \\
\Sigma_2 &= \{v[i,j] : \ 1 \leq i \leq n, \ 1 \leq j \leq k\} \\
\Sigma_3 &= \{p[\alpha], p'[\alpha] : \ \alpha \in I\} \\
\Sigma_4 &= \{q[\alpha, j], q'[\alpha, j] : \ \alpha \in [n]^t, \ 1 \leq j \leq k\} \\
\Sigma_5 &= \{u[\alpha, i, j] : \ \alpha \in [n]^t, \ 1 \leq i \leq n, \ 1 \leq j \leq k\}
\end{aligned}$$

Symbol Subsets, Order and Rank Let Σ' denote $\Sigma_1 \cup \Sigma_2 \cup \Sigma_4 \cup \Sigma_5$. If S_1 and S_2 are sets of symbols of an ordered alphabet, then $S_1 < S_2$ denotes that for all $a \in S_1$ and $b \in S_2$, $a < b$. We consider that Σ' is linearly ordered in the unique way consistent with the following requirements:

- $(\Sigma_1 \cup \Sigma_2) < (\Sigma_4 \cup \Sigma_5)$
- Σ_2 is ordered lexicographically by symbol index.
- For all $i \in [n]$ and $j \in [k]$, $c[j] < v[i,j] < c'[j]$.
- For $1 \leq i < j \leq k$, $\{c[i], c'[i]\} < \{c[j], c'[j]\}$.
- If (α, j) precedes (β, h) lexicographically, then $\{q[\alpha, j], q'[\alpha, j]\} < \{q[\beta, h], q'[\beta, h]\}$.
- Σ_5 is ordered lexicograpically by symbol index.
- $q[\alpha, j] < u[\alpha, i, j] < q'[\alpha, j]$ for all $\alpha \in [n]^t$, $i \in [n]$ and $j \in [k]$.

By $\Sigma'[a,b]$ we denote the set of symbols $\{s \in \Sigma' : a \leq s \leq b\}$ in the above linear ordering.

Each of the symbols in $\Sigma'' = \Sigma_3 \cup \Sigma_4 \cup \Sigma_5$ is (partially) indexed by some $\alpha \in I$. We term the *rank* of a symbol s in this set, denoted $|s|$, to be the rank $|\alpha|$ of the index α. If $, \subseteq \Sigma$ is a set of symbols, then $\Sigma''[r]$ denotes the set of symbols in $,$ of rank r, $0 \leq r \leq t$.

In discussing strings over the alphabet Σ, if $, \subseteq \Sigma$ is a symbol subset and $S \in \Sigma^*$, then by $S \cap ,$ we denote the subsequence of S consisting of all symbols in $,$. We write $|S|$ to denote the length of a string S.

Substring Gadgets Product notation in the description of these components refers to string concatenation. Where s is a symbol, the notation s^w denotes the symbol s repeated w times. Note that in some cases products are formed in decreasing order according to some index, which is indicated by notation such as

$$\prod_{i=n}^{1} \cdots$$

The following strings provide gadgets for our reduction.

$$\langle \uparrow \text{ selection } j \rangle = c[j] \left(\prod_{i=1}^{n} v[i,j] \right) c'[j]$$

$$\langle \downarrow \text{ selection } j \rangle = c[j] \left(\prod_{i=n}^{1} v[i,j] \right) c'[j]$$

$$\langle \uparrow \text{ select } \rangle = \prod_{j=1}^{k} \langle \uparrow \text{ selection } j \rangle$$

$$\langle \downarrow \text{ select } \rangle = \prod_{j=1}^{k} \langle \downarrow \text{ selection } j \rangle$$

As before, where $,$ is a set of symbols, we use $\langle *, * \rangle$ to denote an arbitrary string which contains as a subsequence every string of length m over $,$. As a notational convenience, we write $\langle * s ... t * \rangle$ for $\langle * \Sigma_3 \cup \Sigma'[s,t] * \rangle$.

Recursively, we define $\langle \uparrow \alpha \rangle$ and $\langle \downarrow \alpha \rangle$ for $\alpha \in I$.

For $\alpha \in [n]^t$ and $l[\alpha] = u_i \in V$:

$$\langle \uparrow \alpha \rangle = p[\alpha] \left(\prod_{j=1}^{k} q[\alpha, j] u[\alpha, i, j] q'[\alpha, j] \right) p'[\alpha]$$

$$\langle\downarrow\alpha\rangle = p[\alpha]\left(\prod_{j=k}^{1} q[\alpha,j]u[\alpha,i,j]q'[\alpha,j]\right)p'[\alpha]$$

In general:

$$\begin{array}{rcll}\langle\uparrow\alpha\rangle & = & p[\alpha]^{w(|\alpha|,t)}\prod_{i=1}^{n}\langle\uparrow\alpha.i\rangle p'[\alpha]^{w(|\alpha|,t)} & \\ \langle\downarrow\alpha\rangle & = & p[\alpha]^{w(|\alpha|,t)}\prod_{i=1}^{n}\langle\downarrow\alpha.i\rangle p'[\alpha]^{w(|\alpha|,t)} & \text{for } \alpha \text{ even} \\ \langle\downarrow\alpha\rangle & = & p[\alpha]^{w(|\alpha|,t)}\prod_{i=n}^{1}\langle\downarrow\alpha.i\rangle p'[\alpha]^{w(|\alpha|,t)} & \text{for } \alpha \text{ odd}\end{array}$$

The Reduction We may now describe the reduction. The instance of LCS-1 to which we reduce consists of three sets of strings: the *Control Strings*, the *Quorum Strings* and the *Consistency Strings*.

The two *Control Strings* are

$$X_1 = \langle\uparrow \text{ select } \rangle\langle\uparrow\epsilon\rangle$$

$$X_2 = \langle\downarrow \text{ select } \rangle\langle\downarrow\epsilon\rangle$$

Let $\Delta = \{(r,s) : 1 \leq r < s \leq n\}$ ordered lexicographically. The $\binom{k}{2}$ *Quorum Strings* are, for $1 \leq i < j \leq n$

$$Y_{ij} = \left(\prod_{(r,s)\in\Delta}^{\uparrow\text{lex}} \langle * c[1]\cdots c[i]*\rangle v[r,i]\langle * c'[i]\cdots c[j]*\rangle v[s,j]\langle * c'[j]\cdots c'[k]*\rangle\right)\langle *\Sigma'' *\rangle$$

The $2k$ *Consistency Strings* are, for $j = 1,\ldots,k$

$$Z_j = \prod_{r=1}^{n}\langle \text{ selection } j \text{ is variable } r \rangle$$

$$Z'_j = \prod_{r=n}^{1}\langle \text{ selection } j \text{ is variable } r \rangle$$

where

$$\begin{array}{rcl}\langle \text{ selection } j \text{ is variable } r \rangle & = & \langle * c[1]\cdots c[j]*\rangle v[r,j]\langle * c'[j]\cdots c'[k]*\rangle \\ & & \cdot\left(\prod_{\alpha\in[n]^t}^{\uparrow\ lex}\langle * q[\alpha,1]\cdots q[\alpha,j]*\rangle u[\alpha,r,j]\langle * q'[\alpha,j]\cdots q'[\alpha,k]*\rangle\right)\end{array}$$

Proof of Correctness The following general ideas are useful to our arguments. To the expression E there naturally corresponds a Boolean tree circuit $C = C_E$. A truth assignment τ to the variables V of E may be considered as an input vector x_τ to the circuit C, with $C(x_\tau) = 1$ if and only if τ satisfies E. The circuit C maybe described:
(1) for each $\alpha \in I$, there is a gate g_α of C (of *rank* $|\alpha|$),
(2) g_α is an *and* gate if α is even, and an *or* gate if α is odd,
(3) the output gate of C is g_ϵ,
(4) for $|\alpha| < t$ the gate g_α takes input from the gates $g_{\alpha.i}$ for $i = 1, ..., n$,
(5) the inputs to C are in 1:1 correspondence with V, and
(6) for $|\alpha| = t$, the gate g_α takes the single input $u_i \in V$ such that $l[\alpha] = u_i$ in E.

A subcircuit C' of C is a *witnessing subcircuit* if it satisfies the conditions:
(1) $g_\epsilon \in C'$,
(2) for each even $\alpha \in I$, $|\alpha| < t$, if $g_\alpha \in C'$ then for all $i \in [n]$, $g_{\alpha.i} \in C'$, and
(3) for each odd $\alpha \in I$, if $g_\alpha \in C'$ then there is a unique $i \in I$ such that $g_{\alpha.i} \in C'$.

The following observations about witnessing subcircuits are useful.

Claim 1. $C(x) = 1$ if and only if there is a witnessing subcircuit C' of C such that $C'(x) = 1$ and each gate of C' evaluates to 1.

Claim 1 follows trivially from the monotonicity of C.

Claim 2. If C' is a witnessing subcircuit of C then the number of gates of rank r, for $r = 0, ..., t$, is given by the function

$$c(j) = n^{\lceil j/2 \rceil}$$

Claim 2 follows by an elementary induction, noting the special structure of C.

The following fact about the "weighting function" $w(j,t)$ will be useful.

Claim 3. For $0 \le r \le t-1$,

$$w(r,t) > \sum_{j=r+1}^{t} |X_1 \cap \Sigma''[j]|$$

Claim 3 is easily verified from the definitions.

Claim 4. In the Control Strings X_1 and X_2:
(1) Each symbol in Σ_3 occurs as a *block*, that is, the symbol occurs only in a substring consisting of some number of repetitions of the symbol.
(2) If $\alpha \prec \beta$ then all symbols with index β occur between the block of symbols $p[\alpha]^{w(|\alpha|,t)}$ and the block of symbols $p'[\alpha]^{w(|\alpha|,t)}$.

(3) If β is an index of a symbol occuring between the symbol blocks $p[\alpha]^{w(|\alpha|,t)}$ and $p'[\alpha]^{w(|\alpha|,t)}$ then $\alpha \preceq \beta$, with $\alpha \prec \beta$ properly if the symbol is in Σ_3.

Claim 4 is readily observed from the definition of $\langle \uparrow \alpha \rangle$ and $\langle \downarrow \alpha \rangle$.

In one direction, the argument for the correctness of the reduction is relatively easy. Given a satisfying weight k truth assignment $\tau : V \to \{0,1\}$ for E, we describe a common subsequence of length m in the following way. Let C' be a witnessing subcircuit of C for the input vector corresponding to τ. Let I' denote the set of indices of the logic gates of C'

$$I' = \{\alpha : g_\alpha \in C'\}$$

and suppose the variables set to 1 by τ are $v_{i_1}, ..., v_{i_k}$, with $v_{i_1} < v_{i_2} < \cdots < v_{i_k}$.

Let , denote the set of symbols

$$\begin{aligned} , \; = \; & \Sigma_1 \cup \{v[i_j, j] : 1 \leq j \leq k\} \cup \{p[\alpha], p'[\alpha] : \alpha \in I'\} \\ & \cup \{q[\alpha, j], q'[\alpha, j] : \alpha \in [n]^t \cap I',\ l[\alpha] = v_{i_j},\ 1 \leq j \leq k\} \\ & \cup \{u[\alpha, i_j, j] : \alpha \in [n]^t \cap I',\ l[\alpha] = v_{i_j},\ 1 \leq j \leq k\} \end{aligned}$$

Claim 5. The string $S = X_1 \cap ,$ is a common subsequence of the Control and Consistency Strings of length m.

Proof of Claim 5. First note that $S = S_1 S_2$ where $S_1 \in (\Sigma_1 \cup \Sigma_2)^*$ and $S_2 \in (\Sigma'')^*$ and that similar factorizations hold for X_1 and X_2. An inspection of the definition of $\langle \uparrow$ select $\rangle$ and $\langle \downarrow$ select $\rangle$ shows that S_1 is a common subsequence of the first parts of X_1 and X_2, the main point being that between each pair of symbols $c[j]$ and $c'[j]$ there is just the single symbol $v[i_j, j]$ in S. Note also that the length of S_1 is $3k$.

Let $X'_i = X_i \cap \Sigma''$ for $i = 1, 2$. We argue that S_2 is a subsequence of X'_1 and X'_2 by inducting on the rank of symbols in S_2. Let $S_2[r]$ denote the subsequence of S_2 consisting of the symbols of rank r. By Claim 4, it is sufficient to establish that $S_2[r]$ is a subsequence of X'_1 and X'_2 for $r = 0, ..., t$. The base step of the induction, $r = 0$, is trivial. For the induction step, by Claim 4, it suffices to show that the subsequence of S consisting of symbols with index $\beta = \alpha.i$ (for some i) having rank $r+1$ is a subsequence of X'_i (for $i = 1, 2$) occuring between the symbol blocks $p[\alpha]^{w(|\alpha|,t)}$ and $p'[\alpha]^{w(|\alpha|,t)}$. If α is even then this follows from the fact that the blocks of the symbols $p[\alpha.i]$ and $p'[\alpha.i]$ occur in ascending order in both X'_1 and X'_2. If α is odd then this follows trivially because there is only one relevant index $\alpha.i \in I'$. Note that in S there are precisely 3 symbols between each pair of symbol blocks $p[\alpha]^{w(|\alpha|,t)}$ and $p[\alpha]^{w(|\alpha|,t)}$ where $\alpha \in [n]^t \cap I'$, and that there are $n^{t/2}$ such pairs. From this it is easy to verify that S has length m.

The above arguments establish that S is a subsequence of the Control Strings. By essentially the same inductive argument, the symbols $p[\alpha]$ must occur in S in lexicographically increasing order. Using this fact it is straightforward to verify that S is a subsequence of $\langle$ selection j is variable $i_j\rangle$ and thus S is a subsequence of Z_j and Z'_j for $j = 1, ..., k$. S_2 is trivially a subsequence of $\langle *\Sigma'' * \rangle$. For $p < q$, $v_{i_p} < v_{i_q}$, so S is a subsequence of Y_{pq}. □

To complete the proof of correctness for the reduction, we argue that if T is a common subsequence of the Control and Consistency Strings of length m then E is satisfied by a weight k truth assignment. In particular, we argue that T must correspond to a weight k input vector and a witnessing subcircuit of $C = C_E$ with respect to this vector.

Because the Control Strings can be factored in a similar way, we may factor T as $T = T_1 T_2$ with $T_1 \in (\Sigma_1 \cup \Sigma_2)^*$ and $T_2 \in (\Sigma'')^*$.

Claim 6. The length of T_1 is at most $3k$.

Claim 6 follows simply from the fact that for any fixed index j the symbols $v[i, j]$ occur in X_1 in increasing order with respect to i and they occur in X_2 in decreasing order with respect to i.

Say that an index $\alpha \in I$ is *represented* in T if both of the symbols $p[\alpha]$ and $p'[\alpha]$ occur in T. Say that an index $\alpha \in I$ is *forbidden* in T if for all indices β with $\alpha \preceq \beta$, no symbol with index β occurs in T. The following is an immediate consequence of the definition.

Claim 7. If α is forbidden in T, then for all $i \in [n]$, $\alpha.i$ is forbidden.

Claim 8. If $\alpha \in I$ is odd, then there is at most one $i \in [n]$ with $\alpha.i$ represented in T. Furthermore, if $\alpha.i$ is represented in T, then for all $j \neq i$, $\alpha.j$ is forbidden in T.

Proof of Claim 8. Suppose $i < j$ with $\alpha.i$ represented in T. By the definition of X_1 and X_2, all of the symbols with index $\alpha.i$ precede all of the symbols with index $\alpha.j$ in X_1, and all of the symbols with index $\alpha.i$ succeed all of the symbols with index $\alpha.j$ in X_2. Consequently no symbol with index $\alpha.j$ can occur in the common subsequence T. Furthermore, if β is an index with $\alpha.j \preceq beta$, then by Claim 4, all symbols with index β occur in X_1 and X_2 between blocks of symbols with index $\alpha.j$, so the above argument applies as well to symbols with these indices, so that $\alpha.j$ is forbidden. The case of $j < i$ is symmetric. □

Let $s(r)$ denote the number of indices $\alpha \in I$ of rank r that are represented in T.

Claim 9. For $r = 0, ..., t$
(1) $s(r) = c(r)$
(2) Every index of rank r is either represented or forbidden.

Proof of Claim 9. By induction on r. For $r = 0$, if either $p[\epsilon]$ or $p'[\epsilon]$ fails to occur in T, then necessarily $|T \cap \Sigma''[0]| \leq w(0, t)$, so T must contain at least $w(0, t)$ symbols of rank at least 1, a contradiction of Claim 3. This establishes both (1) and (2).

For the induction step, if $s(r+1) < c(r+1)$ then the induction hypothesis and the definition of m imply that T must contain more than $w(r+1,t)$ symbols of rank greater than $r+1$, which contradicts Claim 3. Suppose $s(r+1) > c(r+1)$.

Case 1: r is even. Then $c(r+1) = n \cdot c(r)$. By (1) of the induction hypothesis, there are precisely $s(r) = c(r)$ indices of rank r represented in T, and all other indices of rank r are forbidden. Since each represented index of rank r has only n extensions to an index of rank $r+1$, there must be some rank $r+1$ index $\alpha.i$ represented in T for which α is not represented in T. By (2) of the induction hypothesis, α is forbidden in T, a contradiction. Thus (1) must hold, and by the same argument, if α of rank r is represented then for all $i \in [n]$, $\alpha.i$ is represented. If α of rank r is forbidden in T, then by Claim 7, $\alpha.i$ is forbidden in T for all $i \in [n]$. This establishes (2).

Case 2: r is odd. Then $c(r+1) = c(r)$. By (1) of the induction hypothesis there are precisely $s(r) = c(r)$ indices of rank r represented in T, and all other indices of rank r are forbidden. There cannot be an index $\alpha.i$ represented in T with α not represented, as this would contradict (2) of the induction hypothesis. By the Pigeonhole Principle, there must be an index α represented in T and $i \neq j$ with both $\alpha.i$ and $\alpha.j$ represented in T, a contradiction of Claim 8. Thus (1) must hold, and by the same arguments we see that for each represented α of rank r there is a unique $i \in [n]$ with $\alpha.i$ represented. By Claim 8 we get (2). □

One can observe from the definition of $\langle \uparrow \alpha \rangle$ and $\langle \downarrow \alpha \rangle$ that there can be at most 3 symbols in $T \cap (\Sigma_4 \cup \Sigma_5)$ with a given index α of rank t, and that these must occur between $p[\alpha]$ and $p'[\alpha]$ and must occur in a substring of the form: $q[\alpha, j]u[\alpha, i, j]q'[\alpha, j]$. By this observation and Claims 6 and 9 we can conclude:

Claim 10. The length of T_1 is precisely $3k$ and the length of T_2 is precisely

$$3n^{t/2} + 2\sum_{j=0}^{t} c(j)w(j,t)$$

On the basis of Claim 10 we may associate to T a well-defined truth assignment τ to the variables of the expression E: $\tau(u_i) = 1$ if and only if for some j, $1 \leq j \leq k$, the symbol $v[i,j]$ occurs in T_1. By Claim 10, there are exactly k symbols of Σ_2 in T_1. However, we must argue that for $j < j'$, only one of $v[i,j]$ and $v[i,j']$ occurs in T_1, thus insuring that τ has weight k. To see this, note that T must be a subsequence of $Y_{jj'}$. Suppose $v[i,j]$ occurs in T (necessarily in T_1). The only symbols $v[i',j']$ occuring in $Y_{jj'}$ after $v[i,j]$ satisfy $i < i'$, by the definition of $Y_{jj'}$. Thus to T we may associate a truth assignment of weight k.

Claim 11. If $\alpha \in [n]^t$ is represented in T, with $u[\alpha, i, j]$ occuring between $p[\alpha]$ and $p'[\alpha]$, then $v[i,j]$ occurs in T_1 (and τ assigns u_i the value 1).

Proof of Claim 11. By Claim 10, there must be a symbol $v[p,j]$ in T_1. The symbol $u[\alpha,i,j]$ occurs only once in Z_j and in Z'_j. The symbols $v[p,j]$ preceding the occurence of $u[\alpha,i,j]$ in Z_j satisfy $p \leq i$. The symbols $v[p,j]$ preceding the occurence of $u[\alpha,i,j]$ in Z'_j satisfy $p \geq i$. Thus the only possibility is $v[i,j]$. □

Claim 12. The indices $\alpha \in I$ represented in T are those of a witnessing subcircuit C' of C that accepts the input vector x_τ corresponding to the truth assignment τ.

Proof of Claim 12. That the indices represented in T form a witnessing subcircuit C' of C follows from Claims 8 and 9. Since C' is monotone, it suffices to establish that all gates of rank t evaluate to 1 in order to conclude that $C'(x_\tau) = 1$. This follows from Claim 11, noting that if α of rank t is represented, then $u[\alpha,i,j]$ occurs between $p[\alpha]$ and $p'[\alpha]$ if and only if $l[\alpha] = u_i$, by the definition of $\langle \uparrow \alpha \rangle$ and Claim 4. □

By the correspondence between C and E, we conclude that τ is a weight k truth assignment for E, which completes the proof of the theorem. □

It is presently not known whether LCS-1 belongs to $W[P]$. The argument given above does not seem to generalize to a proof of $W[P]$-hardness. It is easy to observe that LCS-2 belongs to $W[P]$, by a reduction to whether a circuit C accepts a weight m vector indicating the common subsequence s, where C represents a deterministic P-time computation verifying for each input string X_i that s is a subsequence of X_i.

Theorem 3 *LCS-2 is hard for $W[2]$.*

Proof. We reduce from DOMINATING SET. Let $G = (V,E)$ be a graph with $V = \{1,...,n\}$. We will construct a set S of strings that have a common subsequence of length k if and only if G has a k-element dominating set.

The alphabet for the construction is

$$\Sigma = \{\alpha[i,j] : 1 \leq i \leq k, 1 \leq j \leq n\}$$

We use the following notation for important subsets of the alphabet.

$$\Sigma_i = \{\alpha[i,j] : 1 \leq j \leq n\}$$

$$\Sigma[t,u] = \{\alpha[i,j] : (i \neq t) \text{ or } (i = t \text{ and } j \in N[u])\}$$

The set S consists of the following strings.

Control Strings

$$X_1 = \prod_{i=1}^{k} (\uparrow \Sigma_i)$$

$$X_2 = \prod_{i=1}^{k} (\downarrow \Sigma_i)$$

Check Strings
For $u = 1, ..., n$:

$$X_u = \prod_{i=1}^{k} (\uparrow \Sigma[i, u])$$

To see that the construction works correctly, first note that by Claim 1 of the proof of Theorem 1, it follows easily that any sequence C of length k common to both control strings must consist of exactly one symbol from each Σ_i in ascending order. Thus to such a sequence C we may associate the set V_C of vertices represented by C: if $C = \alpha[1, u_1] \cdots \alpha[k, u_k]$, then $V_C = \{u_i : 1 \le i \le k\} = \{x : \exists i \ \alpha[i, x] \in C\}$.

We argue that if C is also a subsequence of the check strings $\{X_u\}$, then V_C is a dominating set in G. To this end, let $u \in V(G)$ and fix a substring C_u of X_u with $C_u = C$.

Claim. For some index j, $1 \le j \le k$, the symbol $\alpha[j, u_j]$ occurs in the $(\uparrow \Sigma[j, u])$ portion of X_u, and thus $u_j \in N[u]$ by the definition of $\Sigma[j, u]$.

We argue by induction on k. The case of $k = 1$ is clear. For the induction step, there are two cases: (1) the first $k - 1$ symbols of C_u occur in the prefix $(\uparrow \Sigma[1, u]) \cdots (\uparrow \Sigma[k - 1, u])$ of X_u, and the induction hypothesis immediately yields the Claim, or (2) the symbol $\alpha[k - 1, u_{k-1}]$ occurs in the $(\uparrow \Sigma[k, u])$ portion of $C_u \cap X_u$. In case (2), this implies that the symbol $\alpha[k, u_k]$ of $C = C_u$ also occurs in the $(\uparrow \Sigma[k, u])$ part of X_u.

By the Claim, if C is a subsequence of the *Control* and *Check* strings, then every vertex of G has a neighbor in V_C, that is, V_C is a dominating set in G.

Conversely, if $D = \{u_1, ..., u_k\}$ is a k-element dominating set in G with $u_1 < \cdots < u_k$, then the sequence $C = \alpha[1, u_1] \cdots \alpha[k, u_k]$ is easily seen to be common to the strings of S. □

4 Conclusions

Our results suggest that the general LCS problem is not fixed-parameter tractable when either k or m are fixed. It is important to note, however, that our results here apply only to the version of the problem where the size of the alphabet is unbounded. Since many applications involve fixed-size alphabets, the question of whether LCS-1 remains hard for W for a fixed alphabet size is very interesting. We have recently been able to show that LCS remains hard for $W[t]$ for all t when parameterized by both the number of strings and the alphabet size.

Our results also have implications for the fixed-parameter tractability of the multiple sequence alignment and consensus subsequence discovery problems in molecular biology. This is so because the LCS problem is a special case of each of these problems. The problem of aligning k sequences is often re-stated as that of finding a minimal-cost path between two vertices in a particular type of edge-weighted k-dimensional graph [Pev92]. The LCS problem can be stated in this form using the edge-weighting in Section 3 of [Pev92], and is hence a restriction of the multiple sequence alignment problem (albeit, that version of the problem which allows arbitrary alignment evaluation functions). The LCS problem is shown to be a restriction of the consensus subsequence problem in Section 3 of [DM93b]. By the results of this paper, the general multiple sequence alignment (consensus subsequence discovery) problem is W[t]-hard for all t ($W[2]$-hard), and hence unlikely to be fixed-parameter tractable, when the number of sequences and the cost of the alignment (length of the consensus subsequence) are fixed.

Fixed-parameter complexity analysis may be relevant to many computational problems in biology. Many of these problems are known either to be NP-complete in general, e.g. evolutionary tree estimation by parsimony, character compatibility and distance-matrix fitting criteria (see [War93] and references), or to require time $O(n^k)$ when k is fixed, such as multiple sequence alignment using the SP or evolutionary tree alignment evaluation functions [Pev92]. To solve such problems in practice, investigators must often settle for suboptimal solutions obtained by algorithms that are fast but are either approximate or solution-constrained [KS83,San85,Pev92,Gus93,War93]. For instances of such problems, critical parameters such as the number of sequences or taxa are often small but nontrivial, e.g., $5 \leq k \leq 20$. These are precisely the situations in which fixed-parameter tractability might be useful. Apart from showing that for some problems fixed-parameter tractability is unlikely by analyses such as presented in this paper, such results can be viewed as clarifying the contribution that each parameter makes to a problem's complexity. This may suggest computation-saving constraints that may yet yield restricted versions of these problems of feasible complexity.

References

[Bae91] R. A. Baeza-Yates. Searching subsequences. *Theoretical Computer Science* 78 (1991), 363–376.

[BFH94] H. Bodlaender, M. Fellows and M. Hallett. Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy. *Proceedings of the 26th ACM Symposium on the Theory of Computing* (1994), 449–458.

[CCDF93] L. Cai, J. Chen, R. Downey and M. Fellows. The parameterized complexity of short computations and factorization. University of Victoria, Technical Report, Department of Computer Science, July, 1993.

[DEF93] R. Downey, P. Evans and M. Fellows. Parameterized learning complexity. *Proc. Sixth ACM Workshop on Computational Learning Theory (COLT)*, pp. 51–57, ACM Press, 1993.

[DF92] R. Downey and M. Fellows. Fixed-parameter intractability (extended abstract). In *Proceedings of the Seventh Annual Conference on Structure in Complexity Theory*, pp. 36–49, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[DFKHW93] R. Downey, M. Fellows, B. Kapron, M. Hallett and H.T. Wareham. The parameterized complexity of some problems in logic and linguistics. *Proceedings of the Symposium on the Logical Foundations of Computer Science*, Springer Verlag, Lecture Notes in Computer Science, vol. 813 (1994), 89–100.

[DM93a] W. H. E. Day and F. R. McMorris. Discovering consensus molecular sequences. In O. Opitz, B. Lausen, and R. Klar (eds.) *Information and Classification – Concepts, Methods, and Applications*, pp. 393–402, Springer-Verlag, Berlin, 1993.

[DM93b] W. H. E. Day and F. R. McMorris. The computation of consensus patterns in DNA sequences. *Mathematical and Computer Modelling* 17 (1993), 49–52.

[Gus93] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology* 55 (1993), 141–154.

[Hir83] D. S. Hirschberg. Recent results on the complexity of common subsequence problems. In D. Sankoff and J. B. Kruskal (eds.) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pp. 325–330, Addison-Wesley, Reading, MA, 1983.

[IF92] R. W. Irving and C. B. Fraser. Two algorithms for the longest common subsequence of three (or more) strings. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber (eds.) *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching*, pp. 214–229, Lecture Notes in Computer Science no. 644, Springer-Verlag, Berlin, 1992.

[KS83] J. B. Kruskal and D. Sankoff. An anthology of algorithms and concepts for sequence comparison. In D. Sankoff and J. B. Kruskal (eds.) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pp. 265–310, Addison-Wesley, Reading, MA, 1983.

[LF78] S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man, and Cybernetics* 8 (1978), 381–389.

[Mai78] D. Maier. The complexity of some problems on subsequences and supersequences. *Journal of the ACM* 25 (1978), 322–336.

[PM92] W. R. Pearson and W. Miller. Dynamic programming algorithms for biological sequence comparison. *Methods in Enzymology* 183 (1992), 575–601.

[Pev92] P. A. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Mathematics* 52 (1992), 1763-1779.

[San72] D. Sankoff. Matching comparisons under deletion/insertion constraints. *PNAS* 69 (1972), 4–6.

[San85] D. Sankoff. Simultaneous solution of the RNA folding, alignment, and protosequence problems. *SIAM Journal on Applied Mathematics* 45 (1985), 810–825.

[War93] H. T. Wareham. *On the Computational Complexity of Inferring Evolutionary Trees*, M.Sc. Thesis, Technical Report no. 9301, Department of Computer Science, Memorial University of Newfoundland, 1993.