

Edit Distance of Finite State Transducers

C. Aiswarya ✉ 

Chennai Mathematical Institute, India and IRL ReLaX, CNRS France

Amaldev Manuel ✉

Indian Institute of Technology Goa

Saina Sunny ✉ 

Indian Institute of Technology Goa

Abstract

We lift metrics over words to metrics over word-to-word transductions, by defining the distance between two transductions as the supremum of the distances of their respective outputs over all inputs. This allows to compare transducers beyond equivalence.

Two transducers are *close* (*resp.* k -close) with respect to a metric if their distance is finite (*resp.* at most k). Over integer-valued metrics computing the distance between transducers is equivalent to deciding the closeness and k -closeness problems. For common integer-valued edit distances such as, Hamming, transposition, conjugacy and Levenshtein family of distances, we show that the closeness and the k -closeness problems are decidable for functional transducers. Hence, the distance with respect to these metrics is also computable.

Finally, we relate the notion of distance between functions to the notions of diameter of a relation and index of a relation in another. We show that computing edit distance between functional transducers is equivalent to computing diameter of a rational relation and both are a specific instance of the index problem of rational relations.

2012 ACM Subject Classification Theory of computation → Transducers; Theory of computation → Quantitative automata

Keywords and phrases transducers, edit distance, conjugacy

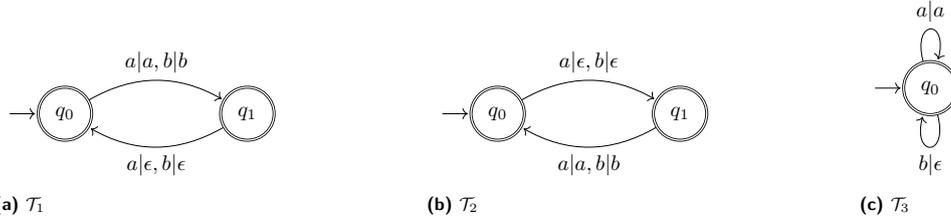
Funding *Amaldev Manuel*: Supported by the DST SERB MATRICS grant MTR/2022/000628 *Deciding closeness of finite state transducers*.

1 Introduction

For meaningfully comparing two words (or sequences, vectors, functions, etc.), it is often necessary to have a measure that quantifies their (dis)similarity. It usually consists of associating a nonnegative integer to two words that indicates how different they are from each other. This usually defines a distance between words, the most popular of which are edit distances. It is the minimum number of edit operations required to transform one word into another. These operations typically include inserting or deleting a letter, substituting a letter with another, swapping adjacent letters (transpositions), and cyclic shifts. Edit distances are studied in coding [31, 44], parsing [2, 35], speech recognition [34, 1], molecular biology [20, 26] etc. Interesting combinatorial problems on words such as the computation of longest common subsequences can be reduced to computing edit distances [6]. For a detailed overview of the history and applications of edit distances, see [29].

The notion of distance between two words can be lifted naturally to distance between a word and a set of words, or between two sets of words, and so on. There is a long line of research of this kind: computing the edit distance between two languages — usually defined as the smallest distance between any two pairs from the respective sets. It could be between a word and a regular language [45] [4], two regular languages [10], a regular language and itself [27], or a regular language and a context-free language [23]. In all these settings there are efficient algorithms for computing the edit distances.

In this paper, we study the distance between two word-to-word functions (transductions) given by finite state transducers, i.e., automata with output. A *transducer* is a machine that reads an input word and produces one or more output words. They were one of the earliest machines to be studied in automata theory, and in fact finite state automata were first defined as a special case of finite state transducers [36]. Finite state transducers are used in a variety of software and hardware systems such as encoders, decoders, demuxers, spell checkers, text normalizers, schema translators, template code generators, etc.



■ **Figure 1** \mathcal{T}_1 outputs letters at the odd positions, \mathcal{T}_2 outputs letters at the even positions and \mathcal{T}_3 outputs only a 's.

Our aim is to develop a framework to meaningfully compare two transductions beyond equivalence. Consider the functions given by the transducers in Figure 1. The transducers \mathcal{T}_1 and \mathcal{T}_2 output the letters at the odd and even positions respectively, while the transducer \mathcal{T}_3 erases b 's in the input. If we were to find the odd one among these three functions, arguably \mathcal{T}_3 will be picked, with the length of the respective output on any input deviating significantly from that of the others. Our aim is to define a measure that quantifies such distances.

If we have a metric to compare the output words, we can extend it to transductions as follows. The distance between two transductions is the least upper bound of the distances between their respective outputs on any input word. We assume that their domains are the same, and we set the distance to infinity if this is not the case. We say that two transductions are close if their distance is finite, and they are k -close if their distance is at most k . We may simply say that two transducers are close (or k -close) instead, to mean that the transductions defined by these transducers are close (or k -close).

We are interested in the following question: *Given two finite state transducers, are the transductions defined by them close (or simply are the transducers close)?* Clearly, deciding closeness is a boundedness problem. We show closeness as well as k -closeness are decidable for various edit distance metrics, in particular Hamming (letter-to-letter substitutions), transposition (swapping adjacent letters), conjugacy (only cyclic shifts) and Levenshtein family of distances — Longest common subsequence (insertion and deletion), Levenshtein (insertion, deletion and substitution), and Damerau-Levenshtein (insertion, deletion, substitution and adjacent transposition). It turns out that computing distance between transducers is equivalent to deciding closeness and k -closeness over integer-valued metrics (see Proposition 3.6). Hence for the edit distances mentioned above, the distance between transducers is computable.

A related notion is that of *diameter* of a relation. We define it to be the supremum of the distance of every pair in the relation. We are interested in computing the diameter of rational relations over words, that is those given by (not necessarily functional) finite state transducers. A rational relation is said to have *bounded diameter* (*resp.* k -bounded diameter) if the diameter of the relation is finite (*resp.* at most k). It turns out that for every pair of transductions \mathcal{T}_1 and \mathcal{T}_2 there is a rational relation R such that for every metric, the diameter of R is same as the distance between \mathcal{T}_1 and \mathcal{T}_2 . In fact, the converse is also true

by virtue of Nivat’s theorem (see Theorem 3.23).

Another related notion is that of the *index of a rational relation in the composition closure of another*. Let R, S be a rational relation over words. The index of R in the composition closure of S is defined to be the smallest integer k such that the relation R is contained in the k -fold composition of S . If such a k exists we say that R has the *finite index property* in the composition closure of S . We show that the finite index property is undecidable for arbitrary rational relations. However, if S is a metrizable relation (see Definition 3.18) w.r.t. the edit distances mentioned above, the index of R in the composition closure of S is computable.

Our decision procedure for k -closeness involves designing a weighted automaton that counts the number of edit operations for transforming one output to the other. We need to check whether there are input instances for which the weight is more than k . We extract a finite state automaton of size exponential in k that achieves this (see Proposition 3.11). This is a generic approach independent of the particular edit operations. However for Hamming and transposition distances, we have a direct polynomial time procedure for deciding k -closeness (see Section 4.3).

Recall that deciding the closeness of transductions is the same as deciding whether the diameter of a rational relation R is bounded. For the latter, consider a transducer recognising R . It turns out that if there are loops in this transducer that produce nonconjugate words (that are not cyclic shifts of each other) then such loops can be iterated to get unbounded diameter/distance. Thus a crucial ingredient in our decision procedure is checking for conjugacy of loops, which is decidable [3]. For boundedness w.r.t. Levenshtein distances, we show that this is also a sufficient condition (see Claim 4.9). For conjugacy distance, we show that the diameter of a rational relation R is bounded if and only if every pair in R is conjugate (see Proposition 4.6). Notice that this is not the case for arbitrary relations. In the case of Hamming distance, which only includes substitutions, we show that it is sufficient to check if the pairs of words generated by the loops after some shifted delay are identical (see Claim 4.11). This also holds true for transposition distance, but additionally, we also need to check if the words are permutations of each other (see Claim 4.12).

1.1 Related Work

The *adjacent functions* in [37] is an analogous definition for closeness between transductions with respect to prefix distance. Two functions $f, g : A^* \rightarrow B^*$ are *adjacent* if $\sup \{d_p(f(w), g(w)) \mid w \in \text{dom}(f) \cap \text{dom}(g)\} < \infty$. Here, $d_p(u, v) = |u| + |v| - 2 \max\{|z| \mid u, v \in zA^*\}$ denotes the *prefix distance* between two words u and v . The adjacency of two rational functions is used in deciding the sequentiality of a function. It is decidable to check if two given rational functions are adjacent or not [37](Proposition 1).

Another problem that is similar in spirit is the *robustness problem*. We say a transducer \mathcal{T} is *robust* w.r.t. a distance d if there is a nontrivial relation R between the distance between two input words (say $d(u, v)$) and distance between their corresponding outputs on \mathcal{T} (say $d(\mathcal{T}(u), \mathcal{T}(v))$). For instance, R could be *Lipschitz continuity* — there is some $k > 0$ such that $d(\mathcal{T}(u), \mathcal{T}(v)) \leq k \cdot d(u, v)$, or *locally Lipschitz continuity* — there exists $b, k > 0$ such that if $d(u, v) < b$ then $d(\mathcal{T}(u), \mathcal{T}(v)) \leq k \cdot d(u, v)$, etc. Sometimes, weaker notions of distance are considered (for instance by dropping the triangle inequality), and respective distances are called *cost* or *similarity* functions. The work [38] solves the locally Lipschitz continuity problem for sequential and unambiguous transducers using reversal bounded counter automata. The problem is shown to be undecidable for Lipschitz continuity even for deterministic transducers and the decidability is shown for the class that has a bound on the

delay between input and output words [25].

Frougny and Sakarovitch studied rational relations with bounded delay [22], which is actually our diameter problem for rational relations when the distance over words is measured by their length difference. A problem related to the diameter of a rational relation is *almost reflexivity* of rational relations studied in [12]. A relation $R \subseteq A^* \times A^*$ is k -reflexive, for some integer $k \leq \infty$, if every element u of the domain is at a distance at most k from some element of the range v , with $(u, v) \in R$, and vice versa. The relation R is almost reflexive if $k < \infty$. It is shown undecidable to check if a deterministic rational relation is almost reflexive, or k -reflexive, for any given integer k , with respect to the following – Hamming, prefix, suffix, subword and Levenshtein edit distances. It is shown decidable for synchronized rational relation w.r.t. Hamming distance.

In 1966, Brzozowski raised the question of *finite power property* on regular languages — it takes a regular language L as input and asks whether there exists some positive integer n such that $(L + \epsilon)^n = L^*$. It was solved in 1979 by Hashiguchi [24] and Simon [40], independently. We study the *finite index property* of a rational relation in the iterative composition of another relation. Notice that the finite index property is different from the finite power property in two respects. One, it is over relations and not languages, and secondly and more importantly, the iteration is obtained by relation composition and not concatenation.

1.2 Organisation of the Paper

In Section 2, we recall the definitions of finite state transducers, metrics on words and edit distances. In Section 3, we define the notion of distance between transducers, the diameter of a rational relation, and the index of a rational relation in another. We also establish the relation between these notions and state our results in this section. In Section 4, we give the connections with conjugacy and the proof arguments remaining from Section 3. Finally, we conclude in Section 5 with a short discussion on future directions.

2 Preliminaries

Let A^* denote the set of all finite words over the alphabet A . We use $|w|$ to denote the length of the word w . Let $w[i..j]$ denote the factor of w from index i to j where $1 \leq i \leq j \leq |w|$. A transduction is a function from words to words.

2.1 Finite State Transducers

The simplest form of a transducer is a deterministic finite state machine whose each transition and each final state is labelled by a possibly empty *output word*. Formally, a *sequential transducer* $\mathcal{T} = \langle \mathcal{A}, \lambda, o \rangle$ with input alphabet A and output alphabet B is a *deterministic finite state automaton* \mathcal{A} with two associated output functions $\lambda : \Delta \rightarrow B^*$ and $o : F \rightarrow B^*$ where Δ and F are the set of transitions and the set of accepting states of \mathcal{A} respectively.

On an input word that is accepted by the automaton, we concatenate the output words produced by the transitions in the unique run of the machine and finally append the end-of-input word of the final state to obtain the output of the machine. That is to say, if $\rho = \delta_1 \cdots \delta_n$ is the successful run of \mathcal{A} on a word $w \in A^*$, the *output* of \mathcal{T} on w , denoted by $\mathcal{T}(w)$, is the word $\lambda(\rho) \cdot o(q)$ where $\lambda(\rho) = \lambda(\delta_1) \cdots \lambda(\delta_n)$ and q is accepting state reached by the run. Let $L(\mathcal{A})$ denote the set of words accepted by \mathcal{A} , called the *language* of \mathcal{A} or the *domain* of \mathcal{T} (denoted as $\text{dom}(\mathcal{T})$). We can see that \mathcal{T} defines a function from $\text{dom}(\mathcal{T})$ to B^* . Functions defined by sequential transducers are called *sequential*. In the literature, they

Edit Distance	Denotation	Allowed Operations
Hamming distance	d_h	letter-to-letter substitutions
Transposition distance	d_t	swapping adjacent letters
Conjugacy distance	d_c	left and right cyclic shifts
Levenshtein edit distance	d_l	insertions, deletions, and substitutions
Longest Common Subsequence	d_{lcs}	insertions and deletions
Damerau-Levenshtein distance	d_{dl}	insertions, deletions, substitutions and adjacent transpositions

■ **Table 1** Edit Distances

are known as *subsequential functions*, introduced by Schützenberger [39]. Transducers given in Figure 1 are sequential.

If we allow the finite state automaton \mathcal{A} to be nondeterministic, then \mathcal{T} no longer defines a function, but a binary relation on $A^* \times B^*$. Such relations are called *rational*. If the relation is a function, then the transducer is called *functional*, and the corresponding functions are called *rational functions*. We can restrict the nondeterminism and still compute all rational functions. A finite state automaton is *unambiguous* if on each input word the machine has at most one run. It is a well-known fact in the theory of transducers that all *rational functions* are computed by finite state transducers whose underlying automata are unambiguous [11]. Such transducers are called *unambiguous transducers*. Clearly sequential functions are a strict subset of rational functions. For instance, the function ‘output the input word if the last letter of the input is an a , otherwise the empty word’ is rational but not sequential.

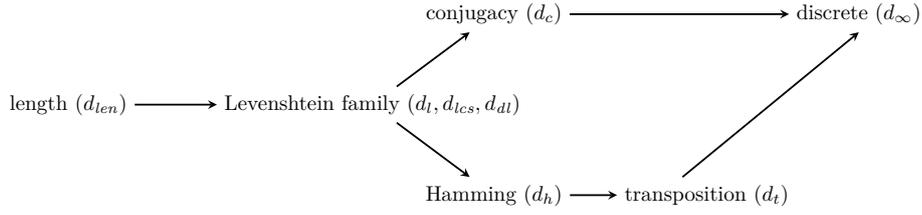
There exist generalisations of rational functions where the underlying automaton is a two-way finite state automaton or equivalently a finite state automaton with registers (corresponding functions are called *regular* [19, 5]), or two-way finite state automaton with pebbles (polyregular functions [8, 9]). An overview of the classical theory of transducers is given in [21]. In this paper, we restrict our attention to one-way functional transducers.

2.2 Metric on Words, Edit Distances

Simply put, a metric on a set is used to measure distance between any two elements of the set. A *metric on words* over the alphabet A is a function $d : A^* \times A^* \rightarrow [0, \infty]$ such that for any words u, v and w in A^* , $d(u, v) = 0 \iff u = v$ (*separation*), $d(u, v) = d(v, u)$ (*symmetry*), and $d(u, v) \leq d(u, w) + d(w, v)$ (*triangle inequality*).

A metric is *integer-valued* if it has range $\mathbb{N} \cup \{\infty\}$. A trivial metric on words is the *discrete metric* — distance between words u and v , denoted by $d_\infty(u, v)$, is 0 if $u = v$ and ∞ otherwise. Another straightforward distance on words is the absolute difference of their lengths (denoted as d_{len}). This is a *pseudo-metric* since the distance between two distinct words can be zero, i.e., does not satisfy the separation property of a metric.

An important class of metrics in the context of word transducers is *edit distances*. Loosely speaking, *edits* are operations that transform words, such as *inserting a letter*, *deleting a letter*, *substitutions (letter-to-letter)*, *adjacent transpositions* (swapping adjacent letters), *left and right shifts* etc. For a fixed set of edit operations C , the edit distance with respect to C between words u and v , is the minimum number of edits in C required to transform u to v if it is possible, and ∞ otherwise. The common edit distances and their corresponding operations are recalled in Table 1. Since many of these operations are obtained by combinations of the others, we can relate these metrics. The notation $d_1 \leq d_2$ is an abbreviation for



■ **Figure 2** The Boundedness preorder of edit distances

$d_1(u, v) \leq d_2(u, v)$ for all words u, v . We can also relate the metrics up to boundedness (See [16] for a detailed introduction). Let $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ be a *correction* function. Usual examples are increments (e.g. $x \mapsto x + 2$), scaling (e.g. $x \mapsto 2 \cdot x$) etc. We extend α to the domain $\mathbb{N} \cup \{\infty\}$ by letting $\alpha(\infty) = \infty$. We write $d_1 \lesssim d_2$ to mean that there is some α such that $d_1 \leq \alpha \circ d_2$. Clearly, if $d_1 \leq d_2$ then $d_1 \lesssim d_2$. If $d_1 \lesssim d_2$ and $d_2 \lesssim d_1$, we write $d_1 \approx d_2$ (this is known as the cost equivalence or the boundedness equivalence). If two functions f and g are cost-equivalent then f and g are bounded over precisely the same family of subsets (See Proposition 1 of [16]).

► **Lemma 2.1.** *The metrics defined in Table 1 are related as follows:*

1. $d_{len} \leq d \leq d_\infty$, for each edit distance metric $d \in \{d_l, d_h, d_t, d_c, d_{lcs}, d_{dl}\}$
2. $d_l \approx d_{lcs} \approx d_{dl}$
3. $d_l \leq d_h \lesssim d_t$
4. $d_l \lesssim d_c$
5. d_c and d_t as well as d_c and d_h are incomparable, i.e., $d_h \not\lesssim d_c, d_c \not\lesssim d_h$ and $d_t \not\lesssim d_c, d_t \not\lesssim d_c$

Proof. 1. From definition of the metrics.

2. Clearly $d_l \leq d_{lcs}$. Since one substitution can be achieved by an insertion and deletion, $d_{lcs} \leq 2 \cdot d_l$. Hence $d_l \approx d_{lcs}$. Similarly $d_{dl} \leq d_l$ and since a transposition is equivalent to two substitutions $d_l \leq 2 \cdot d_{dl}$. Therefore $d_l \approx d_{dl}$.
3. $d_l \leq d_h$ is obvious. Since a transposition is equivalent to two substitutions $d_h \leq 2 \cdot d_t$.
4. Since a cyclic shift is achieved by an insertion and a deletion, $d_l \leq 2 \cdot d_c$.
5. For the last statement, consider the family of words $\{(01)^k, (10)^k \mid k \geq 0\}$, $d_h, d_t \rightarrow \infty$, but d_c is 1. Conversely, for the family of words $\{(10^k 1, 010^{k-1} 1) \mid k \geq 1\}$, $d_t = 1, d_h = 2$, but d_c is ∞ . ◀

3 Distance between Transductions

In this section, we define the notion of distance between two rational functions, the diameter of a rational relation, and the index of a rational relation in another. We establish the relation between these notions and state our results.

3.1 Comparing Transducers

We lift a metric over words to the class of word-to-word functions as follows.

► **Definition 3.1** (Metric on transductions). *Let d be a metric on words over the alphabet B . Given two partial functions $\mathcal{T}, \mathcal{S} : A^* \rightarrow B^*$, we define*

$$d(\mathcal{T}, \mathcal{S}) = \begin{cases} \sup \{ d(\mathcal{T}(w), \mathcal{S}(w)) \mid w \in \text{dom}(\mathcal{T}) \} & \text{if } \text{dom}(\mathcal{T}) = \text{dom}(\mathcal{S}) \\ \infty & \text{otherwise} \end{cases}$$

► **Proposition 3.2.** d is a metric on transductions.

Proof. Clearly, since d is a metric on words, $d(\mathcal{T}, \mathcal{S}) = 0$ if and only if \mathcal{T} and \mathcal{S} are the same transductions, and $d(\mathcal{T}, \mathcal{S}) = d(\mathcal{S}, \mathcal{T})$ for transductions \mathcal{T} and \mathcal{S} . Let $\mathcal{T}, \mathcal{S}, \mathcal{F}$ be three transductions with the output alphabet B . It remains to show that $d(\mathcal{T}, \mathcal{S}) \leq d(\mathcal{T}, \mathcal{F}) + d(\mathcal{F}, \mathcal{S})$.

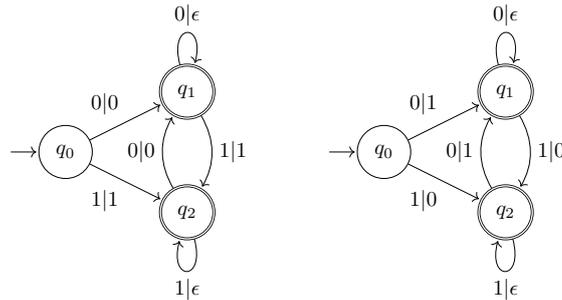
Assume the domains of \mathcal{T} and \mathcal{S} are different. Then, either $\text{dom}(\mathcal{T}) \neq \text{dom}(\mathcal{F})$ or $\text{dom}(\mathcal{F}) \neq \text{dom}(\mathcal{S})$. In both cases, $d(\mathcal{T}, \mathcal{S})$ and $d(\mathcal{T}, \mathcal{F}) + d(\mathcal{F}, \mathcal{S})$ are ∞ . Therefore, assume that the domains of \mathcal{T}, \mathcal{S} and \mathcal{F} are the same, call it L . Since for each word w in L , $d(\mathcal{T}(w), \mathcal{S}(w)) \leq d(\mathcal{T}(w), \mathcal{F}(w)) + d(\mathcal{F}(w), \mathcal{S}(w))$ by virtue of d being a metric, it follows that

$$\begin{aligned} d(\mathcal{T}, \mathcal{S}) &= \sup \{ d(\mathcal{T}(w), \mathcal{S}(w)) \mid w \in L \} \\ &\leq \sup \{ d(\mathcal{T}(w), \mathcal{F}(w)) + d(\mathcal{F}(w), \mathcal{S}(w)) \mid w \in L \} \\ &\leq \sup \{ d(\mathcal{T}(w), \mathcal{F}(w)) \mid w \in L \} + \sup \{ d(\mathcal{F}(w), \mathcal{S}(w)) \mid w \in L \} \\ &= d(\mathcal{T}, \mathcal{F}) + d(\mathcal{F}, \mathcal{S}). \end{aligned}$$

► **Remark 3.** We can define a notion of distance between word-to-word relations in the above manner, however this distance will not be a metric. In particular $d(R, R)$ will not be 0 for a relation R that is not a (partial) function.

► **Example 4.** Consider the sequential transducers \mathcal{T}_1 and \mathcal{T}_2 in Figure 1. The transducers \mathcal{T}_1 and \mathcal{T}_2 output the letters at the odd and even positions respectively. For any input word u , $\| \mathcal{T}_1(u) \| - \| \mathcal{T}_2(u) \| \leq 1$. Hence $d_{\text{len}}(\mathcal{T}_1, \mathcal{T}_2) = 1$. For input word $(ab)^n$ where $n > 1$, the outputs produced by \mathcal{T}_1 and \mathcal{T}_2 are a^n and b^n respectively. Since n substitutions are required to convert a^n to b^n , $d_l(a^n, b^n) = n$. Therefore, $d_h(\mathcal{T}_1, \mathcal{T}_2) = \infty$ as well as $d_l(\mathcal{T}_1, \mathcal{T}_2) = \infty$.

► **Example 5.** The sequential transducer \mathcal{T}_4 in Figure 3 replaces each block of 0's by a single 0 and each block of 1's by a single 1. Similarly, \mathcal{T}_5 substitutes a block of 0's by a single 1 and a block of 1's by a single 0. The output words produced by the transducers on any input word is an alternate sequence of 0's and 1's. If \mathcal{T}_4 outputs 010, then \mathcal{T}_5 produces its complement, i.e., 101. The Hamming distance between \mathcal{T}_4 and \mathcal{T}_5 is ∞ , but the Levenshtein distance is 2.



■ **Figure 3** \mathcal{T}_4 (left) outputs 0 & 1 for each block of 0's & 1's resp. whereas \mathcal{T}_5 (right) outputs 1 & 0 for each block of 0's & 1's resp.

Let d be a distance on words. The value $d(\mathcal{T}, \mathcal{S})$ is an upper bound on how dissimilar the outputs of transducers \mathcal{T} and \mathcal{S} can be on any input. It is natural to ask the computational and boundedness problems given in Table 2.

Closeness and k -closeness are respectively a boundedness and an upper bound problem on distance.

Problem	Input	Question
Distance Problem	transducers \mathcal{T}, \mathcal{S}	$d(\mathcal{T}, \mathcal{S})?$
Closeness Problem	transducers \mathcal{T}, \mathcal{S}	Is $d(\mathcal{T}, \mathcal{S}) < \infty?$
k -Closeness Problem	integer k , transducers \mathcal{T}, \mathcal{S}	Is $d(\mathcal{T}, \mathcal{S}) \leq k?$

■ **Table 2** Problems about distance between two transducers w.r.t. the metric d

► **Proposition 3.6.** *Let d be an integer-valued metric. The distance problem w.r.t. d is computable if and only if k -closeness and closeness problems w.r.t. d are decidable.*

Proof. Clearly, if we can compute the distance w.r.t. d then we can decide k -closeness as well as closeness. For the other direction, given two transducers, we first check if they are close and if it is we perform an exponential search — check if they are k -close for $k = 2^0, 2^1, 2^2, \dots$ till it fails and subsequently perform a binary search on the interval $[2^n, 2^{n+1}]$, $n \in \mathbb{N}$ that contains the distance. ◀

We say two transducers \mathcal{T} and \mathcal{S} are *close* (resp. *k -close*, for $k \geq 0$) w.r.t. d if $d(\mathcal{T}, \mathcal{S}) < \infty$ (resp. $d(\mathcal{T}, \mathcal{S}) \leq k$). Closeness with respect to the discrete metric d_∞ is precisely the equivalence problem. In the case of edit distances, closeness means that the output of \mathcal{T}_1 can be converted to the output of \mathcal{T}_2 by doing a bounded number of edits.

► **Remark 7.** From Definition 3.1, it is easy to verify that Lemma 2.1 holds for transducers as well. If $d_1 \lesssim d_2$, then it is easy to see that if transducers \mathcal{T}_1 and \mathcal{T}_2 are *not* close w.r.t. d_1 , then they are not close w.r.t. d_2 either.

The problems in Table 2 for unambiguous transducers with identical domains can be reduced to that for sequential transducers by considering the cartesian product of the unambiguous transducers. Given two unambiguous transducers \mathcal{T}_1 and \mathcal{T}_2 , we obtain the sequential transducers \mathcal{T}'_1 and \mathcal{T}'_2 as follows. The input automata for \mathcal{T}'_1 and \mathcal{T}'_2 are the same, call it \mathcal{A} , which is the cartesian product of the input automata of \mathcal{T}_1 and \mathcal{T}_2 . By treating the transitions of the cartesian product as the input alphabet, we get input determinism. The output functions of \mathcal{T}'_1 and \mathcal{T}'_2 are lifted from \mathcal{T}_1 and \mathcal{T}_2 respectively.

► **Proposition 3.8.** *Let d be a distance on words. For each pair of unambiguous transducers \mathcal{T}_1 and \mathcal{T}_2 with identical domain, there exist a DFA \mathcal{A} and output functions λ'_1, o'_1 and λ'_2, o'_2 such that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$ where the sequential transducer $\mathcal{T}'_i = \langle \mathcal{A}, \lambda'_i, o'_i \rangle$, $i \in \{1, 2\}$. Furthermore, the size of the automaton \mathcal{A} is polynomial in the size of \mathcal{T}_1 and \mathcal{T}_2 .*

Proof. Assume we are given two unambiguous transducers \mathcal{T}_1 and \mathcal{T}_2 with the input alphabet A and the output alphabet B . Since they are unambiguous, they have at most one accepting run on any input word. The idea is to construct a deterministic automaton whose language is the set of pairs of successful runs of \mathcal{T}_1 and \mathcal{T}_2 on words in their domain.

For convenience, we can assume that \mathcal{T}_1 and \mathcal{T}_2 have exactly one initial state each, denoted by i_1 and i_2 . Let Δ_i, Q_i and F_i , $i \in \{1, 2\}$, be the transitions, the set of states, and the set of final states of \mathcal{T}_i .

The input alphabet of \mathcal{A} is going to be $A' = \Delta_1 \times \Delta_2$. The deterministic automaton \mathcal{A} has the set of states $Q_1 \times Q_2$, initial state (i_1, i_2) , set of final states $F_1 \times F_2$, and the set of transitions (denoted as Δ) of the form $((p, q), (\delta, \chi), (p', q'))$ such that $\delta \in \Delta_1, \chi \in \Delta_2$ are of the form (p, a, p') and (q, a, q') for some letter $a \in A$. It is easy to verify that the language accepted by \mathcal{A} , denoted as $L' \subseteq (\Delta_1 \times \Delta_2)^*$, is the language of all sequences of pairs of the form $\rho = (\delta_1, \chi_1) \cdots (\delta_k, \chi_k)$, $k \geq 1$ and $\delta_i \in \Delta_1, \chi_i \in \Delta_2$ for each $1 \leq i \leq k$, such that

1. δ_i and χ_i are transitions on the same letter,
2. $\delta_1 \cdots \delta_k$ (denoted as $\pi_1(\rho)$) is a successful run of \mathcal{T}_1 ,
3. $\chi_1 \cdots \chi_k$ (denoted as $\pi_2(\rho)$) is a successful run of \mathcal{T}_2 .

Let $\lambda_i : \Delta_i \rightarrow B^*$, $o_i : F_i \rightarrow B^*$, $i \in \{1, 2\}$, be the output functions of \mathcal{T}_i . We define $\lambda'_1 : \Delta \rightarrow B^*$ and $\lambda'_2 : \Delta \rightarrow B^*$ as follows: If $\Delta \ni \psi = ((p, q), (\delta, \chi)(p', q'))$, we let $\lambda'_1(\psi) = \lambda_1(\delta)$ and $\lambda'_2(\psi) = \lambda_2(\chi)$. Similarly, we define $o'_1, o'_2 : F_1 \times F_2 \rightarrow B^*$ as $o'_1(p, q) = o_1(p)$ and $o'_2(p, q) = o_2(q)$, for each $(p, q) \in F_1 \times F_2$.

We take \mathcal{T}'_1 and \mathcal{T}'_2 to be the automaton \mathcal{A} with the output functions λ'_1, o'_1 and λ'_2, o'_2 respectively. Clearly, they are of polynomial size w.r.t. \mathcal{T}_1 and \mathcal{T}_2 . It is easy to show that there is a correspondence (\mapsto) between L and L' , namely $L \ni w \mapsto \rho \in L'$ such that $\pi_1(\rho)$ and $\pi_2(\rho)$ are the unique successful runs of \mathcal{T}_1 and \mathcal{T}_2 on w respectively. Moreover, if $w \mapsto \rho$ then $\mathcal{T}_1(w) = \mathcal{T}'_1(\rho)$ and $\mathcal{T}_2(w) = \mathcal{T}'_2(\rho)$ (\star).

It remains to show that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$. Using (\star) above, we observe that $d(\mathcal{T}_1(w), \mathcal{T}_2(w)) = d(\mathcal{T}'_1(\rho), \mathcal{T}'_2(\rho))$ such that $w \in L, \rho \in L'$ and $w \mapsto \rho$. Since \mapsto is a correspondence, we conclude that $d(\mathcal{T}_1, \mathcal{T}_2) = d(\mathcal{T}'_1, \mathcal{T}'_2)$. \blacktriangleleft

Using the above result, the closeness w.r.t. the length metric d_{len} can be characterised in terms of delay as follows.

► Proposition 3.9. *Consider the sequential transducers $\mathcal{T}_1, \mathcal{T}_2$ defined by the DFA \mathcal{A} and output functions λ_1, o_1 and λ_2, o_2 respectively. The following are equivalent.*

1. $d_{len}(\mathcal{T}_1, \mathcal{T}_2)$ is finite.
2. There is $k \in \mathbb{N}$ such that on any $w \in L(\mathcal{A})$, the difference in lengths of the partial outputs of $\mathcal{T}_1, \mathcal{T}_2$ on any prefix of w is bounded by k .

Proof. (2) \Rightarrow (1) is obvious. For (1) \Rightarrow (2), WLOG assume that automaton \mathcal{A} is trimmed, i.e., all states are accessible (reachable from the initial state) and coaccessible (from each state there is a path to some final state). Let ℓ be the maximum difference in the output lengths on any single transition, and n be the number of states of \mathcal{A} . We claim that, if the difference in output lengths of the transducers on any input is bounded, then $k = n\ell$ validates our proposition. Note that, the difference in the output lengths on any partial input u is at most $|u|\ell$ and we show that this does not exceed k . Suppose $|u|\ell > k = n\ell$, then $|u| > n$, and hence there is a state that is repeated on u such that the repeated part has nonzero difference in the output lengths. That is, $u = u_1u_2u_3$, with $u_2 \neq \epsilon$ such that the states reached after u_1 and that reached after u_1u_2 are the same, say q , and $|\lambda_1(\rho)| - |\lambda_2(\rho)| \neq 0$, where ρ is the unique partial run from q on u_2 . If $w = uv$ for some $w \in L(\mathcal{A})$ then $u_1u_2^i u_3v \in L(\mathcal{A})$ for all i , and the difference in the output lengths become unbounded with increasing i . This contradicts that $d_{len}(\mathcal{T}_1, \mathcal{T}_2)$ is finite. Therefore, the difference in the partial outputs of $\mathcal{T}_1, \mathcal{T}_2$ on any prefix of w is bounded by $k = n\ell$. \blacktriangleleft

Given two transductions \mathcal{T} and \mathcal{S} we define a distance function which maps each word w to the distance between their outputs on w .

► Definition 3.10 (Distance function). *The distance function $f_{\mathcal{T}, \mathcal{S}}^d : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ of \mathcal{T} and \mathcal{S} is*

$$f_{\mathcal{T}, \mathcal{S}}^d(w) = \begin{cases} d(\mathcal{T}(w), \mathcal{S}(w)) & \text{if } w \in \text{dom}(\mathcal{T}) \cap \text{dom}(\mathcal{S}) \\ \infty & \text{otherwise} \end{cases}$$

Transducers \mathcal{T} and \mathcal{S} are close w.r.t. a metric d if their domains are the same and their distance function $f_{\mathcal{T}, \mathcal{S}}^d$ is *limited* (i.e., $< \infty$ on its domain). Similarly k -closeness w.r.t. d of \mathcal{T}

and \mathcal{S} reduces to k -limitedness of $f_{\mathcal{T},\mathcal{S}}^d$. Limitedness problems are well-studied in the context of weighted automata [30, 14]. Therefore, when the distance function $f_{\mathcal{T},\mathcal{S}}^d$ is computable by a $(\min, +)$ -automaton, the distance between \mathcal{T} and \mathcal{S} is computable due to Proposition 3.6.

However, there are distance functions that are not computable by weighted automata. Let $A = \{a, b\}$. Consider the sequential transducers $\mathcal{T}_1, \mathcal{T}_2 : A^* \rightarrow A^*$ with the domain a^*b^* defining the functions $a^pb^q \mapsto a^p$, $a^pb^q \mapsto a^q$ respectively (\mathcal{T}_1 outputs the a 's and erases the b 's, \mathcal{T}_2 erases a 's and renames the b 's as a 's). It is easily checked that their distance function w.r.t. the Levenshtein family ($d \in \{d_l, d_{lcs}, d_{dl}\}$) is $f_{\mathcal{T}_1, \mathcal{T}_2}^d : a^pb^q \mapsto |p - q|$.

If $f : A^* \rightarrow \mathbb{N} \cup \{\infty\}$ is a function computed by weighted automata ($(\min, +)$ or $(\max, +)$ or B -automata [15]), then $L_{f \leq k} = \{w \in A^* \mid f(w) \leq k\}$ is regular for each $k \in \mathbb{N}$. Hence the function $f_{\mathcal{T}_1, \mathcal{T}_2}^d$ is not realised by any of them (consider the language $L_{f_{\mathcal{T}_1, \mathcal{T}_2}^d \leq k}$). In fact, it can be shown that the function $f_{\mathcal{T}_1, \mathcal{T}_2}^d$ is not computed even upto boundedness [17].

To compute k -closeness w.r.t. any of the edit distances, it is not necessary to compute the distance function precisely. The k -approximation of the distance function $f_{\mathcal{T},\mathcal{S}}^d$ is the function $\lceil f_{\mathcal{T},\mathcal{S}}^d \rceil^{\leq k} : w \mapsto f_{\mathcal{T},\mathcal{S}}^d(w)$ if $f_{\mathcal{T},\mathcal{S}}^d(w) \leq k$ and ∞ otherwise.

► **Proposition 3.11.** *If \mathcal{T} and \mathcal{S} are close w.r.t. the length metric, then the approximation $\lceil f_{\mathcal{T},\mathcal{S}}^d \rceil^{\leq k}$ for a metric $d \in \{d_l, d_{lcs}, d_{dl}, d_h, d_t, d_c\}$ is computed by a distance automaton for each $k \in \mathbb{N}$.*

Proof. Let \mathcal{T}_1 and \mathcal{T}_2 be two transducers. The domains of these transducers are identical since they are close w.r.t. the length metric. Hence, we can construct $\mathcal{T}_1 = \langle \mathcal{A}, \lambda_1, o_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}, \lambda_2, o_2 \rangle$ with domain L (by virtue of Proposition 3.8). Since the transducers \mathcal{T}_1 and \mathcal{T}_2 are close w.r.t. the length metric, by virtue of Proposition 3.9, there is a maximum delay $\partial \in \mathbb{N}$ between any partial outputs of \mathcal{T} and \mathcal{S} on any input.

For all metrics $d \in \{d_l, d_h, d_t, d_{lcs}, d_{dl}\}$, fix a set of edits C supported in the metric. We construct a distance automaton $\mathcal{A}_{C,k}$ — weighted automaton over the semiring $(\mathbb{N}, \min, +, \infty, 0)$ — that reads an input word w and accepts it if we can perform at most k edits in C to $\mathcal{T}_1(w)$ and obtain $\mathcal{T}_2(w)$. The cost of a run is the number of edits that it uses. For this the states of automaton $\mathcal{A}_{C,k}$ are those of \mathcal{A} augmented with a budget b and unmatched leftover word of the form (u, ϵ) or (ϵ, u) with $|u| \leq \max(\partial, k)$. Initially, the budget is k , and the word is (ϵ, ϵ) . While performing a transition δ of \mathcal{A} , from a state annotated with b and (ϵ, u) , the automaton $\mathcal{A}_{C,k}$ nondeterministically performs b' edits from C so as to partially match $(\lambda_1(\delta), u\lambda_2(\delta))$. It then moves on to the target state with the reduced budget $b - b'$ and the updated leftover word. The cost of the transition is the number of edits used in the transition, i.e., b' . It is easy to verify that $\mathcal{A}_{C,k}$ exactly computes $\lceil f_{\mathcal{T}_1, \mathcal{T}_2}^d \rceil^{\leq k}$.

For conjugacy distance, we construct a new distance automaton \mathcal{A}_k that remembers a prefix of length at most k of the output of the first (or second) transducer and matches the shifted output with the remaining output of the transducer. The cost of a run is the length of the prefix, i.e., the number of cyclic shifts required to match the outputs. For this, the states of automaton \mathcal{A}_k are those of \mathcal{A} augmented with a word of the form (u, v) with $\|u\| - \|v\| \leq \partial$, and either $\|u\| \leq k$ or $\|v\| \leq k$. Here, u (resp. v) is a suitable candidate for the prefix of the output of the first (resp. second) transducer to be matched with the suffix of the output of the second (resp. first) transducer. Initially the word is (ϵ, ϵ) . While performing a transition δ of \mathcal{A} , from a state annotated with (u, v) , the automaton \mathcal{A}_k either choose to perform step 1 (stores the output of first and second transducer of at most k length), or until nondeterministically chooses to perform step 2 where it starts matching the output of first and second transducer and finally check if the stored output matches with the remaining output.

Problem	Input	Question
Diameter Problem	rational relation R	$dia_d(R)$?
Bounded Diameter Problem	rational relation R	Is $dia_d(R) < \infty$?
k -Bounded Diameter Problem	integer k , rational relation R	Is $dia_d(R) \leq k$?

■ **Table 3** Problems about diameter of a rational relation w.r.t. the metric d

1. Go to the target state with updated word as $(u', v') = (u\lambda_1(\delta), v\lambda_2(\delta))$ only if $\|u' - v'\| \leq \partial$, and $|u'| \leq k$ or $|v'| \leq k$ with the cost of the transition being zero.
2. There are two symmetric cases. Start to partially match the output of the first (*resp.* second) transducer $u\lambda_1(\delta)$ (*resp.* $v\lambda_2(\delta)$) with that of the upcoming output of the second (*resp.* first) transducer, i.e., $\lambda_2(\delta)$ (*resp.* $\lambda_1(\delta)$) only if $|v| \leq k$ (*resp.* $|u| \leq k$). It then moves on to the target state with updated word (ϵ, v) (*resp.* (u, ϵ)) and also the leftover word obtained after matching $u\lambda_1(\delta)$ and $\lambda_2(\delta)$ (*resp.* $v\lambda_2(\delta)$ and $\lambda_1(\delta)$). The cost of the transition is set to be $|v|$ (*resp.* $|u|$). For the upcoming transitions on \mathcal{A} , the matching continues until the output of the second (*resp.* first) transducer is exhausted with cost of transitions being zero. While matching, the state may need to additionally store the leftover words of the form (w, ϵ) or (ϵ, w) . After that, it checks if the stored word v (*resp.* u) matches the first (*resp.* second) transducer's remaining output.

Note that the size of $\mathcal{A}_{C,k}$ and \mathcal{A}_k is exponential in k (as it has to keep track of the unprocessed words). ◀

To check if \mathcal{T} and \mathcal{S} are k -close, we check if they have the same domain and they are close w.r.t. the length metric (otherwise they are neither close nor k -close). If so, we check if the domain of \mathcal{T} is same as the domain of $\lceil f_{\mathcal{T},\mathcal{S}}^d \rceil^{\leq k}$. Thus we get:

► **Corollary 3.12.** *Let d be any metric from Table 1, and \mathcal{T} and \mathcal{S} be any functional transducers. It is decidable if \mathcal{T} and \mathcal{S} are k -close w.r.t. d .*

Interestingly, as elaborated in Section 4.3, for Hamming and transposition distances we have direct procedures for k -closeness that does not involve the weighted automata.

3.2 Diameter of a Rational Relation

► **Definition 3.13** (Diameter of a Rational Relation w.r.t. a distance d). *The diameter of a rational relation R with respect to a distance d , denoted by $dia_d(R)$, is the supremum of the distance of the related words in R .*

$$dia_d(R) = \sup \{ d(u, v) \mid (u, v) \in R \}$$

Similar to the questions asked in Table 2, we can ask the questions given in Table 3 about diameter of a rational w.r.t. a metric d . We say a rational relation has *bounded* (*resp.* *k -bounded*) *diameter w.r.t. a distance d* if the diameter of the relation w.r.t. d is finite (*resp.* $\leq k$). A rational relation with *bounded delay* are precisely those relations with bounded diameter w.r.t. a length metric. Relations with 0-delay are called *length-preserving relations* [18] where any two related words are of equal length. It is decidable to check if a rational relation has bounded delay or 0-delay [22].

Relations bounded w.r.t. the discrete metric are simply those with only identical pairs. It is decidable to determine if a rational relation R is identity. First, check if R is length-preserving. If so, we can construct a letter-to-letter transducer for R based on Eilenberg

Problem	Input	Question
Index Problem	rational relation R, S	$\text{Index}(R, S)?$
Bounded (or Finite) Index Problem	rational relation R, S	Is $\text{Index}(R, S) < \infty?$
k -Bounded Index Problem	integer k , rational relation R, S	Is $\text{Index}(R, S) \leq k?$

■ **Table 4** Problems about the index of a rational relation in the composition closure of another.

and Schützenberger’s theorem [18] stating that a length-preserving rational relation over $A^* \times B^*$ is a rational subset of $(A \times B)^*$, or equivalently, it can be realised by a letter-to-letter transducer whose transitions are labelled with elements of $A \times B$. Finally, validate this transducer for identity by examining the labels of each transition.

3.3 Index of a Rational Relation in a Composition Closure

Consider two rational relations R over $A^* \times B^*$ and S over $B^* \times C^*$. The composition $S \circ R$ over $A^* \times C^*$ is defined by $(S \circ R)(u) = S(R(u)) = \bigcup_{v \in R(u)} S(v)$.

► **Definition 3.14** (Composition closure of a Rational Relation). *Let S be a rational relation over $A^* \times A^*$. Let $S^{(n)}$ denote the composition of S with itself $n \geq 0$ times ($S^{(0)}$ is taken to be the identity relation), and let $S^{\leq(n)}$ denotes the composition of S with itself at most n times, i.e., $S^{\leq(n)} = S^{(0)} \cup S^{(1)} \cup \dots \cup S^{(n)}$.*

The composition closure of S , denoted as $S^{(*)}$, is defined as $S^{(*)} = \bigcup_{i \geq 0} S^{(i)}$.

Notice that we use parenthesis around the superscript to indicate that the base operation is composition, and not concatenation.

► **Definition 3.15** (Index of a Rational Relation in a Composition Closure). *Let S be a rational relation over $A^* \times A^*$. An index of a rational relation R in the composition closure of S , denoted as $\text{Index}(R, S)$, is the smallest integer k such that R is contained in $S^{\leq(k)}$.*

► **Example 16.** *Consider a relation S over $\{a, b\}^* \times \{a, b\}^*$ that deletes the first a if exists on any input. Fix an integer $k > 0$ and let R be the relation that deletes the first k a ’s from the input if exists. The index of R in $S^{(*)}$ is k since for any input word $u \in A^*$, $R(u) \in S^{\leq(k)}(u)$.*

Consider another relation R' that deletes all a ’s from the input. Since $R'(a^{k+1}) \notin S^{\leq(k)}(a^{k+1})$ for any $k > 0$, the index of R' in $S^{()}$ is ∞ .*

As seen in the case of the distance and diameter problem, we can ask questions in Table 4 about the index of a rational relation in the composition closure of a relation. We say a rational relation R has *bounded* (resp. *k -bounded*) index in the composition closure of a rational relation S if the index of R in $S^{(*)}$ is finite (resp. $\leq k$).

The following lemma shows that checking the boundedness of the index problem for an arbitrary rational relation is difficult.

► **Lemma 3.17.** *It is undecidable to check if a rational relation has a bounded index in the composition closure of an arbitrary rational relation.*

Proof. Proof by reducing a version of the halting problem of the Turing machine to the bounded index problem of a rational relation.

A Turing machine is a mathematical model of computation consisting of 7-tuple $M = \langle Q, \Sigma, \Gamma, \delta, q_{in}, q_{accept}, q_{reject} \rangle$ where Q is a finite set of states; $q_{in}, q_{accept}, q_{reject} \in Q$ is the start state, accept state and reject state respectively; Σ, Γ are the finite input and tape alphabet respectively; and δ is the transition function, $\delta : Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \rightarrow$

$Q \times \Gamma \times \{L, R\}$ — given a state and the current tape symbol (pointed by the tape head) the transition writes a new symbol, changes state and moves the tape-head one cell either to the left or to the right. A configuration of a Turing Machine is a snapshot of the current state of the machine. It can be expressed as a word consisting of three components: the tape contents to the left of the tape head, the current state, and the tape contents to the right of the tape head (including the current cell). For example, configuration $x_1x_2 \cdots x_nqy_1y_2 \cdots y_m$, where $x_i, y_i \in \Gamma$, indicates that tape content is $x_1 \cdots x_ny_1 \cdots y_m$ followed by infinite blanks, the tape head is pointing at y_1 and the current state is q . On an input word w , the initial configuration is q_inw , and a configuration of the form $xq_{accept}y$ and $xq_{reject}y$, $x, y \in \Gamma^*$, are called the accepting and rejecting configuration respectively. Let \vdash represent a relation between two configurations C_1, C_2 such that $C_1 \vdash C_2$ only if C_2 is a possible next valid configuration. For instance, for a transition $\delta(q, a) = (q', b, L)$ and current configuration $xqay$, where $x, y \in \Gamma^*, a \in \Gamma$, a next valid configuration would be $xbq'y$. Hence $xqay \vdash xbq'y$. It is easy to verify that \vdash is a rational relation. A computation on a Turing Machine is a sequence of configurations $C_0, C_1 \cdots C_m$ where each configuration is obtained from the previous one by the relation \vdash , i.e., $C_{i-1} \vdash C_i \quad \forall 1 \leq i \leq m$ and C_0 is the initial configuration. If the final configuration, i.e., C_m , is accepting then the word is accepted by the machine (accepting computation). If instead C_m is rejecting then the word is rejected by the machine (rejecting computation). There can be computations where the input word is neither accepted nor rejected and the Turing machine runs forever.

Reduction: First we recall a version of the halting problem for Turing machines.

Input: a Turing machine M and an input word x

Output: Yes, if M halts on x and erases the tape contents; No, otherwise.

Given input M and x , let q_{in} and q_{accept} be the initial and accepting state of M respectively. We can define a singleton relation $R = \{(q_{in}x, q_{accept})\}$. Let S be the \vdash relation over configurations of M . Both R and S are rational relations.

R has a bounded index in $S^{(*)}$ if and only if there exists some $k > 0$ such that $q_{accept} \in \vdash^k(q_{in}x)$. This is equivalent to saying that R has a bounded index in $S^{(*)}$ if and only if M halts on x and erases the tape contents. Since the halting problem for the Turing machine is undecidable, we can conclude that it is undecidable to check if a rational relation has a bounded index in the composition closure of a rational relation. ◀

However, we show that the index problem is decidable w.r.t. a large class of rational relations defined below.

► **Definition 3.18** (Metrizable Relation). *Let S be a rational relation over $A^* \times A^*$. Let $d_S : A^* \times A^* \rightarrow \mathbb{N} \cup \{\infty\}$ be the distance between two vertices in the graph of S , i.e., for any two words u and v , $d_S(u, v)$ is the smallest i such that $v \in S^{(i)}(u)$, and ∞ otherwise.*

We say S is a d -metrizable relation for a metric d if $d_S \approx d$.

► **Proposition 3.19.** *Let R be a rational relation and S be a d -metrizable relation for an integer-valued metric d for which $d_{len} \lesssim d$. If boundedness of diameter w.r.t. d is decidable for a rational relation, then $\text{Index}(R, S)$ is computable.*

Proof. Similar to distance problem, the index problem is computable iff bounded index and k -bounded index problem are decidable. For a rational relation R and d -metrizable relation

S , we show that $\text{Index}(R, S) < \infty$ iff $\text{dia}_d(R) < \infty$ as follows.

$$\begin{aligned}
\text{dia}_d(R) < \infty &\iff \exists k \in \mathbb{N} \text{ s.t. } \forall (u, v) \in R, d(u, v) \leq k \\
&\iff \exists k' \in \mathbb{N} \text{ s.t. } \forall (u, v) \in R, d_S(u, v) \leq k' && \text{(Since } d_S \approx d) \\
&\iff \forall (u, v) \in R, v \in S^{\leq(k')}(u) && \text{(Definition of } d_S) \\
&\iff \text{Index}(R, S) < \infty
\end{aligned}$$

Therefore, if the boundedness of diameter w.r.t. d is decidable for a rational relation, then we can decide if $\text{Index}(R, S) < \infty$. If so, then it suffices to decide if $\text{Index}(R, S) \leq k$ for $k = 0, 1, \dots$ and output the smallest k as the index of R in the composition closure of S .

Since $\text{dia}_d(R) < \infty$ and $d_{\text{len}} \lesssim d$, the rational relation R has a bounded delay. Similarly, S also has a bounded delay since for all $(u, v) \in S$, $d_S(u, v) = 1 \Rightarrow \exists k \in \mathbb{N} \text{ s.t. } d(u, v) \leq k$ (since $d_S \approx d \Rightarrow \exists k' \in \mathbb{N} \text{ s.t. } d_{\text{len}}(u, v) \leq k'$ (since $d_{\text{len}} \lesssim d$). Since S has bounded delay, for any $k \in \mathbb{N}$, $S^{(k)}$ also has bounded delay. It is known that emptiness and set difference of two rational relations with bounded delay is decidable (Corollary 2 of [22]). For any $k \in \mathbb{N}$, deciding $\text{Index}(R, S) \leq k$ reduces to checking if $R \subseteq S^{(k)}$ (or equivalently, $R \setminus S^{(k)} = \emptyset$), and hence decidable. \blacktriangleleft

A close and (almost) dual notion is that of a metric that defines a rational relation.

► **Definition 3.20** (Rationalizable Distance). *A distance d on words is rationalizable if the relation $S_d = \{(u, v) \mid d(u, v) = 1\}$, called the distance relation of d , is rational.*

► **Example 21.** *Consider the hamming distance d_h . We can construct a rational relation $S_h = \{(u, v) \mid u \text{ and } v \text{ differ only in exactly one position}\}$. For example, let $A = \{a, b\}$ and $S_h(aba) = \{bba, aaa, abb\}$. For this, construct a transducer that nondeterministically chooses a position and replaces the input letter with other letters in the alphabet. Similarly, the distance relation of the length metric $S_{\text{len}} = \{(u, v) \mid ||u| - |v|| = 1\}$ is also rational.*

In fact, we have the following result about the rationalizability of edit distances referred in Table 1.

► **Proposition 3.22.** *Every edit distance $d \in \{d_l, d_h, d_t, d_c, d_{lcs}, d_{dl}\}$ is rationalizable.*

Proof. Let d be an edit distance. Let C be the set of permissible edits in d . We can construct a rational relation S such that $S = \{(u, v) \mid d(u, v) = 1\}$. For $d \in \{d_l, d_h, d_t, d_{lcs}, d_{dl}\}$, we can construct a transducer that nondeterministically chooses a position and an edit from set C and applies that edit to the chosen position. For example, the rational relation S_l over $A^* \times A^*$ for Levenshtein edit distance is as follows. For all, $u, v \in A^*$, $(u, v) \in S_l$ if and only if there exists $x, y \in A^*$, $a, b \in A$ with $a \neq b$, such that either one of the following is true.

1. $u = xy, v = xay$ (insertion)
2. $u = xay, v = xy$ (deletion)
3. $u = xay, v = xby$ (substitution)

For conjugacy distance, $S_c = \{(u, v) \mid \text{either } u = ax \text{ and } v = xa, \text{ or } u = xa \text{ and } v = av, x \in A^*, a \in A\}$. For this, we can construct a transducer that nondeterministically remembers (or guesses) the first (or last) letter of the input word and output it at the end (or beginning) along with the rest of the input word. \blacktriangleleft

3.4 Reductions between Distance, Diameter and Index Problems

We show that the distance problem of two rational functions is mutually reducible to the diameter problem of a rational relation, which in turn is mutually reducible to the index problem of a rational relation in the composition closure of a rationalizable distance. Thus, their closeness and boundedness problems are also interreducible.

The correspondence between distance and diameter follows from Nivat's theorem:

► **Theorem 3.23.** ([33]) *Let A and B be alphabets. The following conditions are equivalent.*

1. R is a rational relation over $A^* \times B^*$.
2. There exist an alphabet C , two alphabetic morphisms $\phi : C^* \rightarrow A^*$ and $\psi : C^* \rightarrow B^*$ and a regular language $L \subset C^*$ such that $R = \{(\phi(w), \psi(w)) \mid w \in L\}$

From Proposition 3.8 and (2) \Rightarrow (1) in the above theorem, it follows that distance of two rational functions reduces to the diameter of a rational relation. Now, given a rational relation R , we can create two functional transducers \mathcal{T}_1 and \mathcal{T}_2 in the following way. The domain for these transducers corresponds to the set L in Theorem 3.23. For each transition in \mathcal{T}_1 and \mathcal{T}_2 that involves an input alphabet symbol σ , we set the outputs to be $\phi(\sigma)$ and $\psi(\sigma)$ in Theorem 3.23, respectively. Consequently, \mathcal{T}_1 and \mathcal{T}_2 consist of the sets $\{\phi(w) \mid w \in L\}$ and $\{\psi(w) \mid w \in L\}$ respectively. Since the domain of these transducers is identical, the distance between \mathcal{T}_1 and \mathcal{T}_2 with respect to any distance d , $d(\mathcal{T}_1, \mathcal{T}_2) = \sup \{d(\phi(w), \psi(w)) \mid w \in L\}$, that is equivalent to the diameter of R w.r.t. the distance d .

The correspondence between diameter and index for rationalizable distances is stated in the following proposition.

► **Proposition 3.24.** *The diameter of a rational relation R w.r.t. a rationalizable distance d is equal to the index of the rational relation R in the composition closure of the distance relation of d .*

Proof. Assume that the diameter of a relation R w.r.t. a distance d is ∞ . We claim that the index of R in $S_d^{(*)}$ is also ∞ where S_d is the distance relation of d . Suppose not, i.e., let $k < \infty$ be the index of R in $S_d^{(*)}$. Thus, $\forall (u, v) \in R, v \in S_d^{\leq(k)}(u)$. Since S_d is the distance relation of d , $\forall (u, v) \in R, d(u, v) \leq k$. However, this contradicts the fact that $dia_d(R) = \infty$. Hence, the index of R in $S_d^{(*)}$ is infinite. Similarly, we can prove the other direction. Now, suppose the diameter of R w.r.t. d is finite, i.e.,

$$\begin{aligned} \text{diameter of } R \text{ w.r.t. } d \text{ is } k < \infty &\iff \forall (u, v) \in R, d(u, v) \leq k \\ &\iff \forall (u, v) \in R, v \in S_d^{\leq(k)}(u) \\ &\iff \text{index of } R \text{ in } S_d^{(*)} \text{ is } k. \end{aligned} \quad \blacktriangleleft$$

3.5 Decidability Results

We study the problems stated in Tables 2, 3 and 4 and show that they are decidable for the metrics in Table 1. The index problems stated in Table 4 are undecidable in general (see Lemma 3.17), but is decidable for d -metrizable relations for metrics d given in Table 1.

Recall that, w.r.t. a metric d , distance problem is computable if and only if both closeness and k -closeness are decidable (see Proposition 3.6). We have shown that the k -closeness is decidable for all the metrics in Table 2 (Corollary 3.12). Hence to show the decidability of all the problems in Table 2, it suffices to show the decidability of the closeness problem. Furthermore, thanks to the inter-reductions described above (see Section 3.4), the decidability of Table 3 follows as well as the decidability for the problems in Table 4 for the rationalizable

distance. Moreover, the index of a rational relation in the composition closure of a d -metrizable relation for a metric d given in Table 2 is computable by Proposition 3.19.

It only remains to prove that closeness is decidable for edit distances in Table 1. This is stated below, and proved in the following section.

► **Theorem 3.25.** *Let d be any metric from Table 1, and \mathcal{T} and \mathcal{S} be any functional transducers. It is decidable if \mathcal{T} and \mathcal{S} are close w.r.t. d .*

4 Closeness for Edit Distances

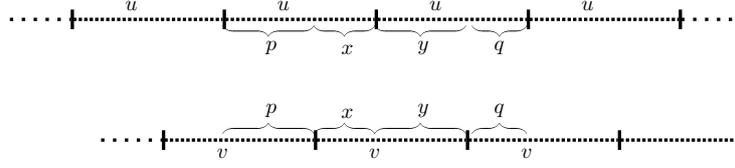
In this section we show that closeness is decidable for all the edit distances in Table 1. The first step is to check if the domain of the transducers are the same. This reduces to checking the equivalence of the underlying automata. For sequential transducers, the underlying automaton is a DFA, while for unambiguous transducers, the underlying automaton is an unambiguous NFA. Checking the equivalence of two unambiguous automata can be done in polynomial time [42], while it is PSPACE in the case of ambiguous automata [43]. Therefore, from now on, we assume that the domains of the transducers given as input to the closeness problem are identical.

Proposition 3.8 allows us to state the distance and closeness problems more abstractly in terms of an automaton over pairs of words. The proposition asserts that distance and closeness problems of two given sequential or unambiguous transducers \mathcal{T}_1 and \mathcal{T}_2 can be reduced to the corresponding problem for a DFA \mathcal{A} with two sets of output functions λ_1, o_1 and λ_2, o_2 . We can combine the output functions to output a pair of words. That is to say, let $\lambda : \Delta \rightarrow B^* \times B^*$ be defined as $\lambda(\delta) = (\lambda_1(\delta), \lambda_2(\delta))$, where $\delta \in \Delta$ and Δ is the set of transitions of \mathcal{A} . Similarly let $o(p) = (o_1(p), o_2(p))$, where $p \in F$ and F is the set of accepting states of \mathcal{A} . Henceforth, we can assume that we are given a DFA \mathcal{A} with the output functions λ and o , denoted as the sequential transducer \mathcal{T} . Since the input words are inconsequential for computing the distance, we can convert the transducer \mathcal{T} to an automaton \mathcal{A} that accepts a set of pairs of output words over $B^* \times B^*$, i.e., $L(\mathcal{A}) = \{(u, v) \in B^* \times B^* \mid (u, v) = \mathcal{T}(w), w \in \text{dom}(\mathcal{T})\}$. Clearly, transducers \mathcal{T}_1 and \mathcal{T}_2 are close w.r.t. d if and only if there exist an integer $k \geq 0$ such that $\forall (u, v) \in L(\mathcal{A}), d(u, v) \leq k$.

Conjugacy of words plays an important role in closeness problems. A pair of words (u, v) is *conjugate* if there exist words x and y (possibly empty) such that $u = xy$ and $v = yx$ or equivalently, u and v are cyclic shifts of one another. For example, $(aaab, aaba)$ is a conjugate pair where $x = a$ and $y = aab$. Conjugacy relation is an equivalence relation on the set of words. A set of pairs is *conjugate* if each pair in the set is conjugate.

► **Lemma 4.1.** *Let \mathcal{T}_1 and \mathcal{T}_2 be two sequential transducers that define a function from A^* to B^* . If \mathcal{T}_1 and \mathcal{T}_2 are close w.r.t. a metric $d \in \{d_l, d_h, d_t, d_c, d_{lcs}, d_{al}\}$, then every loop in the trim automaton over $B^* \times B^*$, that accepts set of all pairs of output words of \mathcal{T}_1 and \mathcal{T}_2 on any input, generates only conjugate pair of words.*

Proof. This proof is an adaptation of a related result in [3]. Let \mathcal{A} be a trim automaton that realises the pair of output words of transducers \mathcal{T}_1 and \mathcal{T}_2 on any input. Since \mathcal{T}_1 and \mathcal{T}_2 are close w.r.t. d , there exist an integer $k \geq 0$ such that $\forall (u, v) \in L(\mathcal{A}), d(u, v) \leq k$. Let (u, v) be a pair labelled in a loop rooted at some state q . Hence (u^ℓ, v^ℓ) for each $\ell \geq 0$ is also a pair in a loop rooted at q . We can safely assume that $|u| = |v|$, otherwise the edit distance will be unbounded as each iteration will increase the edit distance by a difference in length of u and v (Item 1 of Lemma 2.1).



■ **Figure 4** v as infix of uu .

Since \mathcal{A} is trimmed, there exists a path from an initial state q_0 to q and from q to a final state q_f . Let (α_0, β_0) be a pair labelled in a path from q_0 to q , and let (α_1, β_1) be a pair labelled in a path from q to q_f . Thus, pair $(\alpha_0, \beta_0)(u^\ell, v^\ell)(\alpha_1, \beta_1)$ belongs to $L(\mathcal{A})$ where $\ell = 2^k$ (some value much larger than k). Since ℓ is much larger than k and $d(\alpha_0 u^\ell \alpha_1, \beta_0 v^\ell \beta_1) \leq k$, there exist large portions of u 's and v 's that match. Therefore, we can infer that u is a factor of vv , and v is a factor of uu .

Since v is an infix of uu , the following holds as shown in Figure 4. There exist words x, y, p and q such that $v = xy$ and $u = px = yq$. Since $|u| = |v|$, length of p and length of y are the same, that implies $p = y$ (since $u = px = yq$). Therefore, $u = yx$. Hence u and v are conjugate words. Since the pair (u, v) was arbitrary, any pair generated by a loop in \mathcal{A} is conjugate. ◀

4.1 Closeness w.r.t. Levenshtein distances and Conjugacy

In this subsection, we decide closeness w.r.t. Levenshtein family of distances — Levenshtein, Damerau-Levenshtein, and LCS distances — and conjugacy distance. Levenshtein family of distances are all equivalent with respect to closeness problems by Lemma 2.1 and Remark 7.

We have already seen that given two unambiguous transducers \mathcal{T}_1 and \mathcal{T}_2 with identical domains, there exists an automaton \mathcal{A} over $B^* \times B^*$ that accepts a set of all pairs of output words of \mathcal{T}_1 and \mathcal{T}_2 on any input. Thus, we can state the distance and closeness problems in terms of rational expressions over $B^* \times B^*$.

We define *pairs over the alphabet B* to be the set $B^* \times B^*$ with the pointwise concatenation $(u, v) \cdot (u', v') = (u \cdot u', v \cdot v')$. A *rational expression of pairs* over the alphabet B is a rational expression over the alphabet $\{(b, b') \mid b, b' \in (B \cup \{\epsilon\})\}$ that generates subset of pairs over B .

From the automaton \mathcal{A} over $B^* \times B^*$, using state elimination method ([28], Lecture 9), we can construct the rational expression of pairs E for the output pairs generated by the transducer \mathcal{T}_1 and \mathcal{T}_2 on any input. We can lift the metric d to expressions by letting $d(E) = \sup \{d(u, v) \mid (u, v) \in L(E)\}$. Clearly $d(E) = d(\mathcal{T}_1, \mathcal{T}_2)$. Thus, the distance and closeness problems of sequential and unambiguous transducers reduce to the corresponding problems for a rational expression of pairs. Henceforth we assume that we are given a rational expression of pairs.

In the context of conjugacy distance, the closeness of a rational expression necessarily implies that every pair in the expression is conjugate. Otherwise, if there exists a pair $(u, v) \in L(E)$ such that u is not conjugate to v , then $d_c(u, v) = \infty$, thus $d_c(E) = \infty$. In fact, this is also a sufficient condition. The proof relies on the results from [3] that studies the conjugacy of rational expression over pairs of words. It crucially uses the notion of a *common witness* of a set of pairs.

► **Definition 4.2** (Common Witness of a Set of Pairs). *A witness of pair of conjugate words (u, v) is a word z such that either $uz = zv$ (called an inner witness) or $zu = vz$ (called an outer witness). A common witness of a set of pairs is a word z such that either z is an inner*

witness of every pair in the set, or z is an outer witness of every pair in the set.

Lyndon and Schützenberger in 1962 gave a characterisation of conjugacy of a pair of words, stated as a pair of words is conjugate if and only if it has both inner and outer witness (Proposition 1.3.4 of [32]). In [3], it is generalised to a set of pairs as follows.

► **Theorem 4.3.** ([3]) *Let $M = (\alpha_0, \beta_0)G_1^*(\alpha_1, \beta_1) \cdots G_k^*(\alpha_k, \beta_k)$ be a set of pairs where $G_1, \dots, G_k, k > 0$ are arbitrary sets of pairs of words, and $(\alpha_0, \beta_0), \dots, (\alpha_k, \beta_k)$ are arbitrary pairs of words. The set M is conjugate iff M has a common witness.*

Existence of a common witness bounds the conjugacy distance of an expression as follows.

► **Claim 4.4.** *If a rational expression over pairs E has a common witness z , then $d_c(E) \leq |z|$.*

Proof. Since E has a common witness, either $\forall (u, v) \in L(E), uz = zv$, or $\forall (u, v) \in L(E), zu = vz$. WLOG, assume that $\forall (u, v) \in L(E), uz = zv$. Now, for any pair $(u, v) \in L(E)$:

1. If $|u| > |z|$, then z is a prefix of u and suffix of v and hence $(u, v) = (z p, p z)$ for some word $p \in A^*$. Therefore $d_c(u, v) \leq |z|$ since v can be obtained by $|z|$ left cyclic shifts of u .
2. Otherwise, when $|u| \leq |z|$, the number of cyclic shifts required to transform u to v (note that u and v are conjugate since they have a witness) is less than $|u| \leq |z|$. ◀

A rational expression is *sumfree* if it does not use sum (i.e., $+$). In [3], it is shown that if a common witness exists, it is computable for a sumfree rational expression over pairs of words. It is folklore that every rational expression is equivalent to a sum of sumfree expressions [3]. The proposition below implies that to show closeness for a sum of sumfree expressions, it suffices to show closeness for each of its constituent sumfree expressions.

► **Proposition 4.5.** *Let $E = E_1 + \cdots + E_k, k \geq 1$ be a rational expression of pairs. Then $d(E) = \max(d(E_1), \dots, d(E_k))$ for all word metrics d .*

Proof. It suffices to show that if $E = E_1 + E_2$, then $d(E) = \max(d(E_1), d(E_2))$. Since $L(E_i) \subseteq L(E), i \in \{1, 2\}$, we can deduce that $d(E_i) \leq d(E)$ and hence $\max(d(E_1), d(E_2)) \leq d(E)$. It remains to show that $d(E) \leq \max(d(E_1), d(E_2))$. We have two cases.

1. If $d(E) = k \in \mathbb{N}$, then for each pair in E the distance is at most k , and there is a pair $(u, v) \in L(E)$ with distance k . Then, for each pair in E_1 as well as E_2 the distance is at most k , and the expression that contains the pair (u, v) has distance k .
2. If $d(E) = \infty$, then either there is a pair (u, v) with distance ∞ , in which case one of $d(E_1), d(E_2)$ is ∞ . Otherwise, there is an infinite subset of pairs $L \subseteq L(E)$ such that for each $k \in \mathbb{N}$ there is a pair with distance at least k . Since L is infinite, one of $L \cap L(E_1), L \cap L(E_2)$ is infinite, and the corresponding expression has distance ∞ . ◀

An expression is conjugate, if every pair generated by the expression is conjugate. The following proposition characterises closeness w.r.t. conjugacy distance.

► **Proposition 4.6.** *A rational expression over pairs of words is close w.r.t. conjugacy distance if and only if the expression is conjugate. Furthermore, the closeness w.r.t. conjugacy distance is decidable.*

Proof. One direction is trivial. Assume E to be an arbitrary rational expression of pairs and is conjugate. Let $E = E_1 + E_2 + \cdots + E_k$ where E_1, E_2, \dots, E_k are sumfree expressions. Since E is conjugate, each of its sumfree constituent E_i for $1 \leq i \leq k$ is also conjugate. Using Theorem 4.3, each E_i has a common witness, say z_i . From Claim 4.4, $d_c(E_i) \leq |z_i|$. Therefore, $d_c(E)$ is close w.r.t. conjugacy distance by Proposition 4.5. Hence, to decide closeness of E w.r.t. conjugacy distance, it suffices to check if E is conjugate. This reduces to checking if a common witness exists for each sumfree constituents. It is shown to decidable in [3]. ◀

Now consider the case of Levenshtein distances. From Lemma 4.1, if an expression is close w.r.t. Levenshtein distances, it is necessary that every pair generated by a Kleene star in the expression needs to be conjugate. Using common witness, we show that it is also a sufficient condition.

► **Claim 4.7.** *If a rational expression of pairs E has a common witness z , then $d_l(E) \leq 2|z|$.*

Proof. The proof is similar to Claim 4.4. Since E has a common witness, either $\forall(u, v) \in L(E)$, $uz = zv$, or $\forall(u, v) \in L(E)$, $zu = vz$. WLOG, assume that $\forall(u, v) \in L(E)$, $uz = zv$. For any pair $(u, v) \in L(E)$, $|u| = |v|$ since $uz = zv$. There are two cases, either $|u| > |z|$ or $|u| \leq |z|$. If $|u| > |z|$, then z is a prefix of u and suffix of v and hence $(u, v) = (zp, pz)$ for some word p . Therefore, $d_l(E) \leq 2|z|$ by deleting z in the beginning and insert z at the end of u . Suppose $|u| \leq |z|$, the number of edits required to transform u to v is less than $|u| + |v| \leq 2|u| \leq 2|z|$. ◀

► **Proposition 4.8.** *Closeness of a rational expression w.r.t. Levenshtein distance is decidable.*

Proof. Given an arbitrary rational expression, there is an equivalent sum of sumfree expression. From Proposition 4.5, to show closeness for a sum of sumfree expressions, it suffices to show closeness for each of its constituent sumfree expressions. The general form of a sumfree expression $E = (\alpha_0, \beta_0)E_1^*(\alpha_1, \beta_1) \cdots E_k^*(\alpha_k, \beta_k)$ where $k \in \mathbb{N}$, for $0 \leq j \leq k$, (α_j, β_j) is a (possibly empty) pair of words, and for each $1 \leq i \leq k$, E_i is a sumfree expression.

► **Claim 4.9.** *A sumfree expression $E = (\alpha_0, \beta_0)E_1^*(\alpha_1, \beta_1) \cdots E_k^*(\alpha_k, \beta_k)$ is close w.r.t. Levenshtein distance if and only if each E_i^* for $1 \leq i \leq k$ is conjugate.*

Proof. From Lemma 4.1, if E is close w.r.t. Levenshtein edit distance then each E_i^* is conjugate. For the other direction, if each E_i^* is conjugate, then each E_i^* has a common witness, say z_i , by Theorem 4.3. From Claim 4.7, $d_l(E_i^*) \leq 2|z_i|$. Further, $d_l(E) \leq \sum_{j \in \{0 \dots k\}} d_l(\alpha_j, \beta_j) + \sum_{i \in \{1 \dots k\}} d_l(E_i^*) = \sum_{j \in \{0 \dots k\}} d_l(\alpha_j, \beta_j) + 2 \sum_{i \in \{1 \dots k\}} |z_i|$, hence finite. This implies that if each E_i^* in E is conjugate, then $d_l(E)$ is finite. ◀

Therefore, checking the closeness of a rational expression w.r.t. Levenshtein distances reduces to checking the existence of a common witness of each Kleene star in its sumfree constituents, and thus decidable. ◀

For a sumfree rational expression, a witness, if exists, can be computed in polynomial time [3], and thus closeness w.r.t. Levenshtein and conjugacy distances are decidable in polynomial time. However, converting a rational expression to a sum of sumfree rational expressions can cause an exponential blow-up both in the number of summands and the size of each summand [3].

4.2 Closeness w.r.t. Hamming and Transposition distances

► **Theorem 4.10.** *Closeness w.r.t. Hamming and Transposition distance are decidable for functional transducers.*

Given two functional transducers, check if their domains are the same. If not, the distance is ∞ hence they are not close. Assume they have an identical domain. By Proposition 3.8, it suffices to consider two sequential transducers with a common underlying DFA. Let $\mathcal{T}_1 = \langle \mathcal{A}, \lambda_1, o_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}, \lambda_2, o_2 \rangle$ be two sequential transducers. WLOG, we make the following assumptions.

1. (Property \star) Automaton \mathcal{A} is trimmed, i.e., all states are accessible (reachable from the initial state) and coaccessible (from each state there is a path to some final state).
2. (Property \dagger) \mathcal{T}_1 and \mathcal{T}_2 produce output words of identical length: otherwise the Hamming as well as transposition distance will be ∞ . We can check this property: rename all the output letters in \mathcal{T}_1 and \mathcal{T}_2 to a and check their equivalence.
3. The delay between partial outputs of \mathcal{T}_1 and \mathcal{T}_2 is at most $k \in \mathbb{N}$ (By Proposition 3.9).

Let Q and $F \subseteq Q$ be the set of states and final states of \mathcal{A} respectively, and let $q_0 \in Q$ be the initial state. For states $p, q \in Q$, Let $M_{p,q}$ be the set of pairs (u, v) such that there is a run ρ from p to q and $u = \lambda_1(\rho)$ and $v = \lambda_2(\rho)$. Extending this notation, for a state $q_f \in F$, let M'_{q,q_f} be the set of pairs (u, v) such that $u = u' \cdot o_1(q_f), v = v' \cdot o_2(q_f)$ and $(u', v') \in M_{q,q_f}$.

Let q be a state of the automaton. If (α, β) and (α', β') are two pairs in $M_{q_0,q}$, then $|\alpha| - |\beta| = |\alpha'| - |\beta'|$, or else one of the pairs in $\{(\alpha\alpha'', \beta\beta''), (\alpha'\alpha'', \beta'\beta'')\}$ will have different lengths, where (α'', β'') is some pair in M_{q,q_f} , for some $q_f \in F$, guaranteed by Property (\star) . Therefore with each state q , we can associate the delay of a run reaching it, called the *delay at q* , denoted by ∂_q , as $|\alpha| - |\beta|$. Clearly $\partial_q \leq k$. By a symmetric argument, if (α, β) and (α', β') are two pairs in M_{q,q_f} , where q_f is some final state, then $|\alpha| - |\beta| = |\alpha'| - |\beta'| = -\partial_q$. This also implies that for all $(u, v) \in M_{q,q}$, $|u| = |v|$.

For each state q , either $M_{q,q} = \{(\epsilon, \epsilon)\}$, or $M_{q,q}$ is infinite. Let q be a state for which $M_{q,q}$ is nonempty. For a delay $\partial \in \mathbb{Z}$, a pair $(u, v) \in M_{q,q}$ where $n = |u| > \partial$, we define the interior of the pair (u, v) as

$$\text{interior}_{\partial}(u, v) = \begin{cases} (u[1 \dots n - \partial], v[\partial + 1 \dots n]) & \text{if } \partial \geq 0 \\ (u[\partial + 1 \dots n], v[1 \dots n - \partial]) & \text{if } \partial < 0 \end{cases}$$

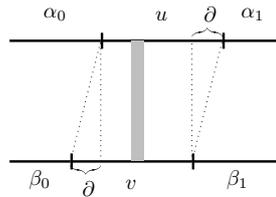
For example, $\text{interior}_1(abc, def) = (ab, ef)$ and $\text{interior}_{-1}(abc, def) = (bc, de)$. We also define the *Left-Border* and *Right-Border* of the pair (u, v) as

$$\text{lborder}_{\partial}(u, v) = \begin{cases} v[1 \dots \partial] & \text{if } \partial \geq 0 \\ u[1 \dots \partial] & \text{if } \partial < 0 \end{cases} \quad \text{rborder}_{\partial}(u, v) = \begin{cases} u[n - \partial + 1 \dots n] & \text{if } \partial \geq 0 \\ v[n - \partial + 1 \dots n] & \text{if } \partial < 0 \end{cases}$$

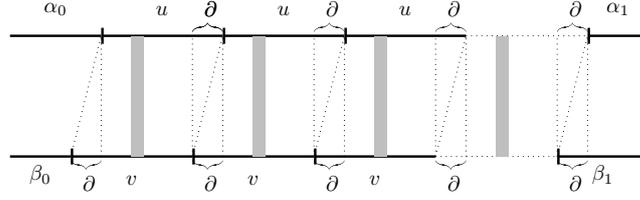
► **Claim 4.11.** *Hamming distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded if and only if there exists a state $q \in Q$ and $(u, v) \in M_{q,q}$ such that $|u| = |v| > \partial_q$, and $u' \neq v'$ where $(u', v') = \text{interior}_{\partial_q}(u, v)$.*

Proof. The Figure 5 depicts the situation described by (2).

(\leftarrow): Assume there exists a state $q \in Q$ and $(u, v) \in M_{q,q}$ such that $|u| = |v| > \partial_q$, and $u' \neq v'$ where $(u', v') = \text{interior}_{\partial_q}(u, v)$. Let $(\alpha_0, \beta_0) \in M_{q_0,q}$ and $(\alpha_1, \beta_1) \in M_{q,q_f}$. Consider the pair $(u_i = \alpha_0 u^i \alpha_1, v_i = \beta_0 v^i \beta_1)$, $i \geq 1$ (shown in Figure 6). Since $u' \neq v'$, we can deduce that $d_h(u_i, v_i) \geq i$. Hence $d_h(\mathcal{T}_1, \mathcal{T}_2) = \infty$.



■ **Figure 5** An edit in the interior of u and v .



■ **Figure 6** Words that require arbitrarily large number of edits.

(\rightarrow): Assume $d_h(\mathcal{T}_1, \mathcal{T}_2) = \infty$. Assume \mathcal{A} has n states and the maximum length of an output produced on any transition or at the end-of-input is ℓ . Choose a run ρ of \mathcal{A} such that the distance between the outputs produced on $\rho = \delta_1 \cdots \delta_m$, $m > 0$ is at least $((k+2)n+1)\ell$. We can associate each edit in $\lambda_1(\rho)$ with the transition δ_i such that the edit happens in $\lambda_1(\delta_i)$. Since there are $((k+2)n+1)\ell$ edits, there are at least $(k+2)n+1$ transitions in ρ whose output words are edited. Associate each transition with its source state. By pigeonhole principle, there is a state q such that $\rho = \rho_1 \cdot \rho_2 \cdot \rho_3$ where

1. ρ_1 is a run from the initial state q_0 to q ,
2. ρ_2 is a run from q to itself,
3. ρ_3 is a run from q to a final state q_f , and
4. there are at least $(k+1)$ edits in the factor $\lambda_1(\rho_2)$.

Let $u = \lambda_1(\rho_2)$ and $v = \lambda_2(\rho_2)$. Clearly $|u| = |v|$ and $|u| \geq (k+1)$. Since the edits in u are at least $k+1$, there is a position on which the pair $\text{interior}_{\partial_q}(u, v)$ differ. ◀

Deciding Claim 4.11: For checking characterisation in Claim 4.11, for each state q for which $M_{q,q}$ is nonempty, we need to check if the $\text{interior}(q) = \{\text{interior}_{\partial_q}(u, v) \mid (u, v) \in M_{q,q}\}$ consist of only identical pairs of words. Towards this, we assume that the transducers given are normalised, i.e., for any transition δ in \mathcal{A} , $\lambda_i(\delta)$ for $i = \{1, 2\}$ is either a single letter or ϵ .

We can verify and compute the unique delay, if exists, for each state in polynomial time. For this, set the delay at initial state to be zero. Now for each transition, compute the delay of the target state. If the delay of the target state is not yet computed, set this as its delay. On the other hand, if delay of target state is already computed, then check the computed delay is equal to that. If not, the delay is not unique and thus, Property (\star) is not satisfied and transducers are not close. Otherwise continue with the next transition of \mathcal{A} till all transitions are covered.

We construct a forward and backward gadget for each state that unfolds the loop up to its delay in both directions trimming the *lborder* and *rborder* respectively. For a state q with the delay ∂_q , if $\partial_q > 0$, then the forward gadget of q will produce pairs $\{(u[1.. \partial_q], \epsilon) \mid (u, v) \in M_{q,q}, |u| > \partial_q\}$, and the backward gadget of q will produce pairs $\{(\epsilon, v[(n - \partial_q) .. n]) \mid (u, v) \in M_{q,q}, n = |v| > \partial_q\}$. Symmetrically for $\partial_q < 0$.

Instead of constructing these gadgets for each state, we combine the construction for each strongly connected component (SCC). In polynomial time, we can find all the strongly connected components (SCCs) of the automaton \mathcal{A} . Let S be the set of states in a SCC and ∂ be the maximum absolute value of delay among all the states in S . Construct a forward and backward gadget (or automaton) \mathcal{F}_S and \mathcal{B}_S respectively with states from $S \times \{-\partial, -\partial+1, \dots, 0, 1, 2, \dots, \partial\}$ with output functions $\lambda_{f_1}, \lambda_{f_2}$ and $\lambda_{b_1}, \lambda_{b_2}$ respectively.

Initially, the forward gadget has states of the form (q, ∂_q) for $q \in S$, and the backward gadget has states of the form $(q, -\partial_q)$ for $q \in S$.

For a transition $\delta = (p, a, q)$ in \mathcal{A} where $a \in A$ and $p, q \in S$,

- In forward gadget, for each state $(p, i) \in \mathcal{F}_S$, we add a transition $\delta_f = ((p, i), a, (q, j))$ such that
 1. if $i > 0$, then $\lambda_{f_1}(\delta_f) = \lambda_1(\delta)$, $\lambda_{f_2}(\delta_f) = \epsilon$, and if $\lambda_2(\delta) \neq \epsilon$ then $j = i - 1$; else $j = i$.
 2. if $i < 0$, then $\lambda_{f_1}(\delta_f) = \epsilon$, $\lambda_{f_2}(\delta_f) = \lambda_2(\delta)$, and if $\lambda_1(\delta) \neq \epsilon$ then $j = i + 1$; else $j = i$.
- In backward gadget, for each state $(q, i) \in \mathcal{B}_S$, we add a transition $\delta_b = ((p, j), a, (q, i))$ such that
 1. if $i > 0$, then $\lambda_{b_1}(\delta_b) = \lambda_1(\delta)$, $\lambda_{b_2}(\delta_b) = \epsilon$, and if $\lambda_2(\delta) \neq \epsilon$ then $j = i - 1$; else $j = i$.
 2. if $i < 0$, then $\lambda_{b_1}(\delta_b) = \epsilon$, $\lambda_{b_2}(\delta_b) = \lambda_2(\delta)$, and if $\lambda_1(\delta) \neq \epsilon$ then $j = i + 1$; else $j = i$.

We construct such a forward and backward gadget for each SCCs. For each state q in an SCC with a set of states S , we construct an automaton \mathcal{A}_{S_q} by concatenating \mathcal{F}_S , \mathcal{A} induced with states S , and \mathcal{B}_S with existing transitions and labels. Additionally, following transitions are added.

- For each state $(p, 0) \in \mathcal{F}_S$, $p \in \mathcal{A}$, add $((p, 0), \epsilon, p)$ with ϵ output labels.
- For each state $p \in \mathcal{A}$, $(p, 0) \in \mathcal{B}_S$, add $(p, \epsilon, (p, 0))$ with ϵ output labels.

Let $\lambda'_1 = \lambda_{f_1} \cup \lambda_1 \cup \lambda_{b_1}$ and $\lambda'_2 = \lambda_{f_2} \cup \lambda_2 \cup \lambda_{b_2}$ be the output labels in \mathcal{A}_{S_q} , and o'_1, o'_2 maps each state to ϵ in \mathcal{A}_{S_q} . Let the initial state of \mathcal{A}_{S_q} being (q, ∂_q) and final state being $(q, -\partial_q)$. It is easy to see that the automaton \mathcal{A}_{S_q} is unambiguous and is of polynomial size w.r.t. \mathcal{A} . To verify Claim 4.11, it suffices to check if the functional transducers $\langle \mathcal{A}_{S_q}, \lambda'_1, o'_1 \rangle$ and $\langle \mathcal{A}_{S_q}, \lambda'_2, o'_2 \rangle$ are equivalent for each state q in a SCC with a set of states S .

Next we show closeness w.r.t. transposition distance. We write $u \equiv v$ to denote that words u and v are permutations of each other. The *alphabetic vector* of a word over the alphabet A , denoted by \vec{u} , is the sequence $(|w|_{a_i})_{a_i \in A}$ for some fixed ordering of A . It is easy to observe that two words are permutations of each other if their alphabetic vectors are the same.

► **Claim 4.12.** *Transposition distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded if and only if one of the following holds*

1. There is a pair $(u, v) \in M'_{q_0, q_f}$, $q_f \in F$ such that $u \not\equiv v$.
2. There exists a state $q \in Q$ and $(u, v) \in M_{q, q}$ such that $|u| = |v| > \partial_q$, and $u' \neq v'$ where $(u', v') = \text{interior}_{\partial_q}(u, v)$.
3. There exists a state $q \in Q$ such that $M_{q, q}$ is infinite, and for each pair $(u, v) \in M_{q, q}$ of length at least $|\partial_q|$, $\text{interior}_{\partial}(u, v)$ is identical. Further, there are pairs $(u, v) \in M_{q, q}$ and $(\alpha, \beta) \in M_{q_0, q}$ (resp. M_{q, q_f}) such that: If $\partial_q \geq 0$, then $\alpha \not\equiv \beta \cdot \text{lborder}(u, v)$ (resp. $\text{rborder}(u, v) \cdot \alpha \not\equiv \beta$), and if $\partial_q < 0$, then $\alpha \cdot \text{lborder}(u, v) \not\equiv \beta$ (resp. $\alpha \not\equiv \text{rborder}(u, v) \cdot \beta$).

Proof. (\Leftarrow): It is obvious that if Item 1 is true, then the transposition distance between \mathcal{T}_1 and \mathcal{T}_2 is unbounded. Therefore we assume that the output pairs of the transducers are permutations of each other. For Item 2, the proof is the same as in Claim 4.11. Next we consider Item 3. The cases are symmetric. Assume that there exist a pair $(u, v) \in M_{q, q}$, $(\alpha, \beta) \in M_{q_0, q}$, and WLOG $\partial_q \geq 0$ such that $\alpha \not\equiv \beta \cdot \text{lborder}(u, v)$. Let (α', β') be some pair in M_{q, q_f} . Consider the pair $(u_i = \alpha u^i \alpha', v_i = \beta v^i \beta')$, $i \geq 1$.

Let $(x, x) = \text{interior}_{\partial_q}(u, v)$, $z_1 = \text{lborder}(u, v)$, $z_2 = \text{rborder}(u, v)$. By assumption $\alpha \not\equiv \beta z_1$, and hence $z_2 \alpha' \not\equiv \beta'$. Since interior of (u, v) is (x, x) , we can deduce that $\alpha z_2 \alpha' \equiv \beta z_1 \beta'$. Therefore $\vec{\alpha} - \vec{\beta} z_1 = z_2 \vec{\alpha}' - \vec{\beta}'$. This means that the transpositions have to cancel out the differences in the vectors at each end of the word. We can prove by induction that it requires at least $|x|$ transpositions to mitigate a difference of 1, while keeping the alphabetic vector of the middle portion the same. Hence we deduce that $d_t(u_i, v_i) \geq i$.

(\rightarrow): If $d_t(\mathcal{T}_1, \mathcal{T}_2) \in \infty$, either there is a pair of outputs (u, v) such that $d_t(u, v) = \infty$ (This is Item 1), or all the output pairs are permutations of each other and there is an infinite set of pairs $S = \{(u_i, v_i) \mid i > 0\}$ such that $d_t(u_i, v_i) \geq i$.

In the latter case, we show that either Item 2 or Item 3 holds. We say the set S is *error-bounded* if there is an $r > 0$ such that u_i and v_i differ in at most r positions. Clearly, there are sets with bounded errors on which d_t is infinite. We do case analysis.

If there is an infinite set of pairs $S = \{(u_i, v_i) \mid i > 0\}$ such that $d_t(u_i, v_i) \geq i$ that is *not* error-bounded, we proceed as in the proof of Claim 4.11 and obtain Item 2 by pigeonhole principle.

If the set of all output pairs is error-bounded, then clearly for states q such that $M_{q,q}$ is infinite, the interior of all the sufficiently large pairs in $M_{q,q}$ are identical. Moreover since the output pairs are permutations of each other there is a state q such that $|M_{q,q}| = \infty$ and there is a partial run from q_0 to q (or a partial run from q to q_f) whose output words are not permutations of each other. \blacktriangleleft

Deciding Claim 4.12: The Item 2 in Claim 4.12 is same as that of characterisation in Claim 4.11, and thus can be checked in polynomial time. For checking Item 3, it suffices to check if all the paths between any two strongly connected components in \mathcal{A} have the same alphabetic vectors. Towards this, first topologically sort the strongly connected components in \mathcal{A} . Consider any two adjacent SCCs with a set of states S_1 and S_2 , respectively in the topological sort of \mathcal{A} with directed paths from states in S_1 to that of S_2 . Our aim is to check if all the paths from a state in S_1 to a state in S_2 have the same alphabetic vector. For this, we construct a induced subgraph of \mathcal{A} , say \mathcal{A}_I , with $S_1 \cup S_2$ being the set of states, along with all edges associated with it. Let S'_1 be the states in S_1 that has an outgoing edge to next SCC in \mathcal{A}_I and S'_2 be the set of states in S_2 that has an incoming edge from previous SCC in \mathcal{A}_I . Assume that the induced graph is trimmed, i.e., any state is reachable from a state in S'_1 and co-reachable from a state in S'_2 . For each state in the induced subgraph, we maintain a vector/ sequence of length $|A|$ for some fixed ordering of A where A is the alphabet of \mathcal{A} such that each position in the vector tells the difference in the count of a letter encountered till now. It can either be a positive value or a negative value. For example, for $A = \{1, b\}$, $(2 \quad -3)$ means there are 2 a 's more (or less) in the first (or second) component of the output and there are 3 b 's less (or more) in the first (or second) component of the output seen till now. Initially, for all states in S'_1 in \mathcal{A}_I we associate a zero vector. For each transition in \mathcal{A} involving states in \mathcal{A}_I , we compute the vector of the target state from the computed vector of the initial state. For any state, if there are two different vectors associated, then since \mathcal{A}_I is trimmed, there exist a path from S'_1 to S'_2 with different alphabetic vector. Thus, for each state in \mathcal{A}_I , there is a unique vector. Finally, we need to check if the vector associated with each state in S'_2 is a zero vector. Otherwise, there exist a path from S'_1 to S'_2 with different alphabetic vector. Hence, Item 3 can be checked in polynomial time. It is easy to see that if both Item 2 and Item 3 are not satisfied, then Item 1 is also not satisfied.

Therefore, Claim 4.11 and Claim 4.12 can be verified for \mathcal{T}_1 and \mathcal{T}_2 in polynomial time. Thus, closeness of sequential and unambiguous transducers w.r.t. hamming and transposition distance is decidable in polynomial time.

4.3 k -closeness of Hamming and Transposition distances

We have shown that using Proposition 3.11, k -closeness is decidable for all metrics in Table 1. To check if two transducers \mathcal{T}_1 and \mathcal{T}_2 are k -close, we check if they have the same domain and they are close w.r.t. the length metric (otherwise they are neither close nor k -close). If

so, we check if the domain T is same as the domain of $[f_{\mathcal{T}_1, \mathcal{T}_2}^d]^{\leq k}$. From the construction used in Proposition 3.11, the underlying automaton of $f_{\mathcal{T}_1, \mathcal{T}_2}^{\leq k}$ can be of exponential (as it has to keep track of the unprocessed words). So, the complexity of the k -closeness algorithm is EXPSpace. However, for hamming and transposition distance, using the characterisations used for deciding their closeness (see Claim 4.11 and Claim 4.12), we give a co-NP procedure for deciding k -closeness and co-NP \cap NP procedure for computing the distance.

► **Theorem 4.13.** *k -closeness w.r.t. Hamming and Transposition distance are decidable for sequential and unambiguous transducers in co-NP time.*

Proof. Given two unambiguous transducers, check in polynomial time that if they are close w.r.t. Hamming (or Transposition) distance. If not, then the distance is ∞ and they are not k -close. Assume they are close.

By Proposition 3.8, it suffices to consider two sequential transducers with a common underlying DFA. Let $\mathcal{T}_1 = \langle \mathcal{A}, \lambda_1, o_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{A}, \lambda_2, o_2 \rangle$ be two sequential transducers.

Since \mathcal{T}_1 and \mathcal{T}_2 are close, from Condition (2) in Claim 4.11 and Condition 2.(b) in Claim 4.12, we get that for any state q in \mathcal{A} , $interior(q)$ produces only identical pair of outputs. In fact, we can ignore the $interior$ of the loops, and by only keeping $lborder$ and $rborder$ of any loops, we construct a polynomially sized acyclic automaton from \mathcal{A} such that distance of \mathcal{T}_1 and \mathcal{T}_2 equals to the maximum over the minimum edits required in any accepting path of the acyclic automaton. Towards this, we construct a forward and backward gadget similar to the one constructed for deciding Claim 4.11. However, in this case, instead of trimming the $lborder$ (*resp.* $rborder$) in the forward (*resp.* backward) gadget, we keep only $lborder$ (*resp.* $rborder$) and make the other component ϵ . We construct such a forward and backward gadget for each SCCs in \mathcal{A} . For an SCC, we combine the forward and backward gadgets by merging the states of the form $(p, 0)$ in both of these gadgets where p is a state in the SCC. This combined gadget is acyclic.

Given \mathcal{A} , we topologically sort the strongly connected components. We replace each SCC with its combined forward and backward gadget. For any transition (p, a, q) between two adjacent SCC, we add a transition from $((p, -\partial_p), a, (q, \partial_q))$ connecting the combined gadgets of the two SCCs. Therefore, we get a polynomially sized acyclic automaton, say \mathcal{A}_T , from \mathcal{A} by removing the $interior$ of any state in \mathcal{A} and keeping the initial and final states the same as that of \mathcal{A} . Now, it suffices to check if all paths from initial to the final state in the acyclic automaton require only at most k edits (substitutions in the case of Hamming distance, while adjacent transpositions in the case of Transposition distance) to convert the output of the first transducer to that of second. The complement of this problem is asking if there exists a path that requires at least $k + 1$ edits to convert one output to another. This is in NP since given a certificate, we can polynomially compute the number of edits required to convert one output to another (for hamming - $\mathcal{O}(n)$ and for transposition - $\mathcal{O}(n \log n)$ [13] where n is the length of the output words), and hence verify polynomially that whether it requires $k + 1$ edits. Since the complement of this problem is in NP, the problem is in co-NP. ◀

As a consequence, we get the following result.

► **Proposition 4.14.** *Computing distance of two transducers w.r.t. Hamming as well as transposition distance is in co-NP \cap NP.*

Proof. First, determine the closeness in polynomial time. If not close, then distance is ∞ . If close, then we can construct an polynomially sized acyclic automaton \mathcal{A}_T such that the distance equals to the maximum over the minimum edits required in any accepting path. In fact, the distance is bounded by product of the number of transitions in \mathcal{A}_T and the

maximum value in the output length on any single transition. Let b denote the bound on the distance. Now, we can compute the exact distance using k -closeness in two ways. One way, is to iterate k over $\{0, 1, \dots, b\}$, and check for each k that every accepting path in \mathcal{A}_T requires at most k edits (which is in co-NP). If not, increment k and continue. Else, k is the distance between T_1 and T_2 . Hence, computing distance is in co-NP. The other way is to iterate k over $\{b, b-1, \dots, 1\}$, and for each k , check if there exists a accepting path with at least k edits (which is in NP). If yes, then k is the required distance between T_1 and T_2 . Otherwise, decrement k and continue. Hence, computing distance is also in NP. ◀

5 Discussion and Conclusion

It is shown that distance between two rational functions w.r.t. common edit distances is computable. The related notions of diameter of a rational relation, and the index of a rational relation in the composition closure of another are also computable. We leave open the question of finding the precise computational complexity of the problems in Tables 2, 3 and 4.

The current decision procedure for closeness w.r.t. conjugacy and Levenshtein family of distances proceeds through the analysis of rational expressions. One could directly work on automata, but it is not enough to check for the conjugacy of simple cycles, as there can be complex strongly connected components. In such cases, a decidability proof for conjugacy can be achieved by utilizing Simon's factorization forests [41] and checking the conjugacy of the factorization trees inductively. Sumfree expressions are doing this in essence, circumventing the need to construct the transition monoids.

Lifting these notions to infinite words, and two-way transducers is an immediate next step. Distance between one-way transducers could be seen as the diameter of a rational relation obtained by the cartesian product. However, when the transducers \mathcal{T}, \mathcal{S} are two-way or polyregular, the relation $\{(\mathcal{T}(w), \mathcal{S}(w)) \mid w \in \text{dom}(\mathcal{T})\}$ need not be rational. It remains to develop techniques for checking the conjugacy of non-rational relations.

An interesting question is: given two functional transducers \mathcal{T}_1 and \mathcal{T}_2 with bounded distance, does there exist a transducer \mathcal{T} such that \mathcal{T}_2 is equivalent to a cascading composition of \mathcal{T}_1 and \mathcal{T} ? This is often called the *repair problem* and is well-studied between two regular languages [7].

References

- 1 M Ackroyd. Isolated word recognition using the weighted levenshtein distance. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(2):243–244, 1980.
- 2 Alfred V Aho and Thomas G Peterson. A minimum distance error-correcting parser for context-free languages. *SIAM Journal on Computing*, 1(4):305–312, 1972.
- 3 C Aiswarya, Amaldev Manuel, and Saina Sunny. Deciding conjugacy of a rational relation. *arXiv preprint arXiv:2307.06777*, 2023.
- 4 Cyril Allauzen and Mehryar Mohri. Linear-space computation of the edit-distance between a string and a finite automaton. In *London Algorithmics 2008: Theory and Practice*. 2008.
- 5 Rajeev Alur and Pavol Černý. Expressiveness of streaming string transducers. In *FSTTCS 2010*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–12, Dagstuhl, Germany, 2010.
- 6 Alberto Apostolico and Concettina Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2:315–336, 1987.

- 7 Michael Benedikt, Gabriele Puppis, and Cristian Riveros. Regular repair of specifications. In *2011 IEEE 26th Annual Symposium on Logic in Computer Science*, pages 335–344. IEEE, 2011.
- 8 Mikolaj Bojanczyk. Transducers of polynomial growth. In *Proceedings of the 37th annual acm/ieee symposium on logic in computer science*, pages 1–27, 2022.
- 9 Mikolaj Bojanczyk, Sandra Kiefer, and Nathan Lhote. String-to-String Interpretations With Polynomial-Size Output. In *ICALP 2019*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 106:1–106:14, Dagstuhl, Germany, 2019.
- 10 Horst Bunke. Edit distance of regular languages. *Proceedings of 5th Annual Symposium on Document Analysis and Information Retrieval*, 01 1996.
- 11 Christian Choffrut and Serge Grigorieff. Uniformization of rational relations. In *Jewels are Forever: Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 59–71. Springer, 1999.
- 12 Christian Choffrut and Giovanni Pighizzini. Distances between languages and reflexivity of relations. *Theoretical Computer Science*, 286(1):117–138, 2002.
- 13 Vincent A Cicirello. Kendall tau sequence distance: Extending kendall tau from ranks to sequences. *arXiv preprint arXiv:1905.02752*, 2019.
- 14 Thomas Colcombet. On factorisation forests. *arXiv preprint cs/0701113*, 2007.
- 15 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *International Colloquium on Automata, Languages, and Programming*, pages 139–150. Springer, 2009.
- 16 Thomas Colcombet. Regular cost functions, part i: logic and algebra over words. *Logical Methods in Computer Science*, 9, 2013.
- 17 Thomas Colcombet, Denis Kuperberg, Amaldev Manuel, and Szymon Toruńczyk. Cost functions definable by min/max automata. In *33rd International Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 1–36, 2016.
- 18 Samuel Eilenberg. *Automata, languages, and machines*. Academic press, 1974.
- 19 Joost Engelfriet and Hendrik Jan Hooeboom. Mso definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, 2(2):216–254, April 2001.
- 20 David Eppstein, Zvi Galil, and Raffaele Giancarlo. Efficient algorithms with applications to molecular biology. In *Sequences: Combinatorics, Compression, Security, and Transmission*, pages 59–74. Springer, 1990.
- 21 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19, aug 2016.
- 22 Christiane Frougny and Jacques Sakarovitch. Rational relations with bounded delay. In *STACS 91: 8th Annual Symposium on Theoretical Aspects of Computer Science Hamburg, Germany, February 14–16, 1991 Proceedings 8*, pages 50–63. Springer, 1991.
- 23 Yo-Sub Han, Sang-Ki Ko, and Kai Salomaa. Computing the edit-distance between a regular language and a context-free language. volume 24, pages 85–96, 08 2012.
- 24 Kosaburo Hashiguchi. A decision procedure for the order of regular events. *Theoretical Computer Science*, 8(1):69–72, 1979.
- 25 Thomas Henzinger, Jan Otop, and Roopsha Samanta. Lipschitz robustness of finite-state transducers. *Leibniz International Proceedings in Informatics, LIPIcs*, 29, 03 2014.
- 26 Richard M Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 278–285, 1993.
- 27 Stavros Konstantinidis. Computing the edit distance of a regular language. *Information and Computation*, 205:1307–1316, 09 2007.
- 28 Dexter Kozen. *Automata and computability*. Undergraduate texts in computer science. Springer, 1997.
- 29 Joseph B Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237, 1983.

- 30 Hing Leung and Viktor Podolskiy. The limitedness problem on distance automata: Hashiguchi's method revisited. *Theoretical Computer Science*, 310(1-3):147–158, 2004.
- 31 Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.
- 32 Roger C Lyndon, Marcel-Paul Schützenberger, et al. The equation $am = bn$ in a free group. *Michigan Math. J*, 9(4):289–298, 1962.
- 33 Maurice Nivat. Transductions des langages de chomsky. In *Annales de l'institut Fourier*, volume 18, pages 339–455, 1968.
- 34 Teruo Okuda, Eiichi Tanaka, and Tamotsu Kasai. A method for the correction of garbled words based on the levenshtein metric. *IEEE Transactions on Computers*, 100(2):172–178, 1976.
- 35 Giovanni Pighizzini. A parallel minimum distance error-correcting context-free parser. In *Proc. 4th Italian Conf. on Theoretical Computer Science*, pages 305–317. World Scientific, 1992.
- 36 Michael Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 04 1959.
- 37 Christophe Reutenauer and Marcel-Paul Schutzenberger. Minimization of rational word functions. *SIAM Journal on Computing*, 20(4):669–685, August 1991.
- 38 Roopsha Samanta, Jyotirmoy V. Deshmukh, and Swarat Chaudhuri. Robustness analysis of string transducers. In *Automated Technology for Verification and Analysis*, pages 427–441, Cham, 2013. Springer International Publishing.
- 39 Marcel Paul Schuetzenberger et al. Sur une variante des fonctions séquentielles. *Theor. Comput. Sci.*, 4(1):47–57, 1977.
- 40 Imre Simon. Limited subsets of a free monoid. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pages 143–150. IEEE Computer Society, 1978.
- 41 Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.
- 42 Stearns and Hunt. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SICOMP: SIAM Journal on Computing*, 14, 1985.
- 43 L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th Symposium on Theory of Computing*, pages 1–9, 1973.
- 44 J Ullman. Near-optimal, single-synchronization-error-correcting code. *IEEE Transactions on Information Theory*, 12(4):418–424, 1966.
- 45 Robert Wagner. Order- n correction for regular languages. *Commun. ACM*, 17:265–268, 05 1974.