Hamming Distance Between Finite Transducers

Luc Dartois
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F- 25000 Besançon, France

Pierre-Cyrille Héam
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F- 25000 Besançon, France

Ismaël Jecker
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F- 25000 Besançon, France

Silvio Vescovo
Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F- 25000 Besançon, France

Abstract

We study bounded deviation of non-deterministic finite transducers under the Hamming distance: the bounded comparison problem asks, given two transducers and $k \in \mathbb{N}$, whether for every input the two transducers produce words at Hamming distance at most k. This problem is known to be decidable in polynomial time when k is fixed, and in co-NP otherwise.

We show that the problem is NL-complete when k is fixed, co-NP-complete when k is given in binary, and it is DP-complete to decide if the distance is exactly k. We also prove that if the two transducers have bounded comparison, then the maximal distance is at most quadratic in the size of both transducers, and that this bound is asymptotically tight.

We prove the results on deviation problems, which asks similar questions on the distance of the pairs of input and output of a single transducer, and show that these two families of problems are logspace many-one equivalent.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases Transducers, Hamming distance, NL-completeness, DP-completeness

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.47

Related Version *Full Version*: `https://arxiv.org/abs/2604.25398`

Funding This research was partially funded by the Agence Nationale de la Recherche (ANR) grant ANR-25-CE48-2447 FAVOR.

Acknowledgements We kindly thank the reviewers for their helpful remarks.

1 Introduction

Non-deterministic finite-state transducers (NFT) are a fundamental model for describing transformations between words. Studied since the early days of computer science, these machines, initially known as *generalized sequential machines* [17, 6], are obtained by equipping transitions of finite-state automata with output words. Whereas an automaton $\mathcal{A}$ recognizes a *language* $L_{\mathcal{A}}$, a transducer $\mathcal{T}$ recognizes a *binary relation* $R_{\mathcal{T}}$ between input and output words, called a *rational relation*. We refer to [14, 5] and the references therein for a comprehensive overview of this model. As is standard in automata theory, classical decision problems on transducers are inherently Boolean (e.g., equivalence, or determinisability). To move beyond this qualitative setting towards quantitative questions, we need to determine a meaningful notion of *distance between transducers*. A key requirement is that such a notion should correspond to algorithmically tractable decision problems to support effective analysis. Lifting distances from words to transducers yields a natural candidate satisfying these requirements.

A common way of comparing two words u and v is through their *edit distance* $d(u, v)$, defined as the minimum number of elementary operations, called *edits*, required to transform u into v. Allowing different edits induce different distances. The most well-known, called *Levenshtein distance* [12], allows insertions, deletions and substitutions, and numerous variants arise by restricting operations or assigning them weights. In this work, we focus on the *Hamming distance* [7], where the only operation allowed is the substitution of a letter by another. Beyond words, edit distances have been extended to richer structures. The distance between two languages is typically defined as the minimal [9] or average [13] distance between pairs of words drawn from each language. More recently, edit distances have been lifted to rational relations [1, 4]. In this setting, the perspective shifts: rather than witnessing proximity via the existence of a close pair, uniform closeness is required among all pairs of outputs associated with the same input. Formally, given two relations R_1 and R_2, their distance is defined as ∞ if their domains are distinct, and otherwise we let

$$d(R_1, R_2) = \sup \{d(v_1, v_2) \mid (u, v_1) \in R_1 \text{ and } (u, v_2) \in R_2 \text{ for some } u \in \Sigma^*\} \in \mathbb{N} \cup \{\infty\}.$$

This worst-case viewpoint is well-suited to quantitative verification, as it can capture guarantees on the deviation from an ideal behavior. This distance between relations leads naturally to three fundamental decision problems: determining whether the distance is finite, whether it is bounded by a given threshold, and whether it is exactly equal to a given value.

Bounded Comparison Problem
Input: Two NFTs $\mathcal{T}_1, \mathcal{T}_2$ with $\mathrm{dom}(R_{\mathcal{T}_1}) = \mathrm{dom}(R_{\mathcal{T}_2})$.
Output: TRUE if $d(R_{\mathcal{T}_1}, R_{\mathcal{T}_2}) < \infty$, FALSE otherwise.

Threshold-bounded Comparison Problem
Input: Two NFTs $\mathcal{T}_1, \mathcal{T}_2$ with $\mathrm{dom}(R_{\mathcal{T}_1}) = \mathrm{dom}(R_{\mathcal{T}_2})$, and $k \in \mathbb{N}$ encoded in binary.
Output: TRUE if $d(R_{\mathcal{T}_1}, R_{\mathcal{T}_2}) \leqslant k$, FALSE otherwise.

Exact Comparison Problem
Input: Two NFTs $\mathcal{T}_1, \mathcal{T}_2$ with $\mathrm{dom}(R_{\mathcal{T}_1}) = \mathrm{dom}(R_{\mathcal{T}_2})$, and $k \in \mathbb{N}$ encoded in binary.
Output: TRUE if $d(R_{\mathcal{T}_1}, R_{\mathcal{T}_2}) = k$, FALSE otherwise.

Restricting the inputs to transducers with identical domains is natural in this setting. Indeed, if $\mathrm{dom}(R_{\mathcal{T}_1}) \neq \mathrm{dom}(R_{\mathcal{T}_2})$, then the distance $d(R_{\mathcal{T}_1}, R_{\mathcal{T}_2})$ is infinite by definition, making the comparison trivial. Moreover, this restriction allows us to isolate the intrinsic complexity of the comparison problems. Without it, one would first need to check whether the domains coincide, which is PSPACE-complete [21] and would therefore dominate the overall complexity.

Contributions and organization of the paper. In this paper we approach these problems from a different perspective: instead of comparing the outputs of two relations we compare the input and output of a single relation. Formally, given a binary relation R, the *deviation* of R is the maximal Hamming distance between input and output over all pairs in R:

$$\mathsf{dev}(R) = \sup \{d(u, v) | (u, v) \in R\} \in \mathbb{N} \cup \{\infty\}.$$

This notion naturally gives rise to the following decision problems.

Bounded Deviation Problem
Input: An NFT $\mathcal{T}$.
Output: TRUE if $\mathsf{dev}(R_{\mathcal{T}}) < \infty$, FALSE otherwise.

Threshold-bounded Deviation Problem
Input: An NFT $\mathcal{T}$ and $k \in \mathbb{N}$ encoded in binary.
Output: TRUE if $\mathsf{dev}(R_{\mathcal{T}}) \leqslant k$, FALSE otherwise.

Exact Deviation Problem
Input: An NFT $\mathcal{T}$ and $k \in \mathbb{N}$ encoded in binary.
Output: TRUE if $\mathsf{dev}(R_{\mathcal{T}}) = k$, FALSE otherwise.

We first show that these problems are equivalent to their comparison counterparts.

▶ **Theorem 1.** *The Bounded, Threshold-bounded and Exact Deviation Problems are logspace many-one equivalent to the corresponding Comparison Problems.*

Theorem 1 is proved in Section 3, where we show how to construct, from two transducers $\mathcal{T}_1, \mathcal{T}_2$, a single transducer $\mathcal{T}$ such that $\mathsf{dev}(R_{\mathcal{T}}) = d(R_{\mathcal{T}_1}, R_{\mathcal{T}_2})$ (Proposition 9), and conversely how to construct, from a transducer $\mathcal{T}$, a pair of transducers $\mathcal{T}_1, \mathcal{T}_2$ such that $\mathsf{dev}(R_{\mathcal{T}}) = d(R_{\mathcal{T}_1}, R_{\mathcal{T}_2})$ (Proposition 10). We then establish tight complexity bounds for all problems.

▶ **Theorem 2.** *The Bounded Deviation Problem for the Hamming distance is NL-complete.*

▶ **Theorem 3.** *The Threshold-bounded Deviation Problem for the Hamming distance is:*
- *co-NP-complete when k is part of the input;*
- *NL-complete for every fixed $k \geqslant 1$, when k is an outside constant.*

The DP complexity class, for *difference polynomial time* is first defined in [16, S. 2] as the class of the problems that are expressed as the difference between two NP problems. Alternatively, it consists of problems that can be defined as the intersection of an NP problem and a co-NP problem. Note that DP contains both NP and co-NP.

▶ **Theorem 4.** *The Exact Deviation Problem for the Hamming distance is DP-complete.*

Note that, by Theorem 1, these results immediately extend to the Comparison Problems. Theorems 2-4 are proved over two sections: in Section 4, we prove the matching hardness results by reductions from canonical complete problems (Propositions 11, 12, 13 and 14). In Section 5, we define algorithms establishing membership of the Bounded Problems in the corresponding complexity class (Propositions 15 and 24). Remark that the Exact Deviation Problem is DP as it is the intersection of the Threshold Bounded Problem which is co-NP, and its complement [22]. We also show a quadratic bound on the size of the transducer for the deviation of bounded NFT.

▶ **Theorem 5.** *For every NFT $\mathcal{T}$, if $\mathsf{dev}(R_{\mathcal{T}}) < \infty$ then $\mathsf{dev}(R_{\mathcal{T}}) = O(|\mathcal{T}|^2)$. Moreover, there exists a family $(\mathcal{T}_n)_{n\in\mathbb{N}}$ such that each $\mathcal{T}_n$ has $2n$ states, $3n-1$ atomic transitions, and satisfies $\mathsf{dev}(R_{\mathcal{T}_i}) = \frac{n^2+n}{2}$.*

The upper bound is shown in Proposition 15. The lower bound is a consequence of Lemma 8.

Related work. The Comparison Problems for both Hamming and Levenshtein distance were already studied in [1], with additional details in the long version [2] and in the PhD thesis of S. Sunny [22]. In particular, it is shown that for the Hamming distance:
1. the Bounded Comparison Problem is decidable in polynomial time [1, Theorem 4.10],
2. the Threshold-bounded Comparison Problem is in co-NP [2, Theorem 4.13],
3. the Exact Comparison Problem is in DP [22, Theorem 6.19].

These results correspond to the upper bounds we revisit in Section 5. Our contributions strengthen them as follows. For the Bounded problem, while we use a similar proof structure and characterization, we provide alternative proofs, and we show that the problem is actually in NL (Proposition 15). Moreover, we establish a tight quadratic bound on the distance when it is finite (Theorem 5). Regarding this bound, although no explicit statement appears in [2], a polynomial bound can be extracted from the proofs therein. We make this bound explicit and show that it is optimal. For the Threshold-bounded Problem, we identify the true source of intractability: the parameter k, rather than the transducers themselves. More precisely, we show that the Threshold-bounded Comparison problem drops from co-NP to NL when k is fixed (Proposition 24). Note that these similarities with [1] concern only Section 5, whereas Section 4 contains entirely new results.

Other extensions of edit distance to rational relations have been considered.

The notion studied in this paper is inherently *universal*: for every input, *all* corresponding outputs must be close. In contrast, the notion of *almost reflexivity*, introduced in [3], adds an existential quantifier: for every input, there must exist *at least one* corresponding output that is close. This notion is less well-behaved algorithmically, as most related decision problems are undecidable over the class of rational relations.

The *robustness* introduced in [19, 10] proposes another way of comparing transducers based on edit distances. Rather than comparing outputs corresponding to a fixed input, like the Comparison Problems, or comparing input and output, like the Deviation Problems, it enforces a Lipschitz-like condition, requiring that close inputs yield close outputs. Again, this notion is algorithmically less well-behaved and leads to undecidability in general: restrictions to subclasses of rational relations are required to recover decidability.

2 Preliminaries

Words and Hamming distance. An *alphabet* is a finite set of symbols called *letters*. A *word* over a given alphabet Σ is a finite sequence of letters of Σ. The *empty word*, denoted ε, is the empty sequence. The set of all words over an alphabet Σ is the free monoid Σ^*. Given a word u, we denote by $|u|$ the length of the sequence of letters of u, i.e. the number of letters in u, and for each i in $1 \leqslant i \leqslant |u|$ we denote by u_i the i-th letter of u. For two words $u = u_1u_2\cdots u_n$, $v = v_1v_2\cdots v_m$ we denote uv the concatenation of u and v: $uv = u_1\cdots u_nv_1\cdots v_m$. Note that the concatenation operation is associative and ε is the neutral element for this operation. For a given word $u \in \Sigma^*$, we say that a word $x \in \Sigma^*$ is a *factor* of u if there exist two words v and w of Σ^* such that $u = vxw$. A *prefix* (resp. *suffix*) $x \in \Sigma^*$ of $u \in \Sigma^*$ is a factor of u where v (resp. w) is the empty word. A word $v_1\cdots v_n$ is a *sub-word* of a word u if there exist some words $w_0, \ldots, w_n$ such that $u = w_0v_1w_1\cdots v_nw_n$. We say that two words u and v are *conjugate* if there exist a prefix w of u and a prefix z of v such that $u = wz$ and $v = zw$. Furthermore, we say that u is *conjugate to v by $n \in \mathbb{Z}$* if u and v are in fact conjugate and for all $i, j \in [1, |u|]$ such that $j - i \equiv n \bmod |u|$, $u_i = v_j$[1]. In this article, we are interested in deciding the similarity between machines and their word outputs, using the *Hamming distance* [8, S. 3.6]. To this end, we define the function d from $(u, v) \in \Sigma^* \times \Sigma^*$ to $d(u, v) \in \mathbb{N} \cup \{+\infty\}$ where:

$$d(u,v) = \begin{cases} +\infty, & \text{if } |u| \neq |v| \\ |\{i \in [1,|u|] \mid u_i \neq v_i\}| & \text{if } |u| = |v| \end{cases}$$

Note that the function d is equal to the Hamming distance when $|u| = |v|$.

[1] Note that it is symmetric, if u is conjugate to v by $0 \leqslant n < |u|$, then v is conjugate to u by $|u| - n$

Non-deterministic Finite-state Transducers. A *non-deterministic finite-state transducer* (NFT) $\mathcal{T}$ is an extension of a finite automaton, i.e. a quintuplet $(Q, \Sigma, Q_i, Q_f, \Delta)$ where Q is a finite set of states, Σ is both the input and output alphabet, $Q_i \subseteq Q$ and $Q_f \subseteq Q$ are respectively the sets of initial and final states. The set of transitions Δ is a finite subset of $Q \times \Sigma^* \times \Sigma^* \times Q$. Note that generally the input and output alphabets are defined as different. However, this definition is without loss of generality as one can always consider Σ to be the union of the input and output alphabets.

A *run* ρ of $\mathcal{T}$ is a word $\delta_1\delta_2\cdots\delta_n$ in Δ^* such that for all $1 \leqslant i < n$, $\pi_4(\delta_i) = \pi_1(\delta_{i+1})$, *where* $\pi_j(\delta)$ *is the projection on the j-th component of* δ. If $n \geqslant 1$, we say that ρ is a run from a state $p = \pi_1(\delta_1)$ to a state $q = \pi_4(\delta_n)$ over the pair of words (u, v), where $u = \pi_2(\delta_1)\pi_2(\delta_2)\cdots\pi_2(\delta_n)$ and $v = \pi_3(\delta_1)\pi_3(\delta_2)\cdots\pi_3(\delta_n)$. The length of a run ρ, denoted by $|\rho|$, is the number of transitions of ρ. A run ρ is said to be *empty* if and only if $|\rho| = 0$, in this case, the run ρ is a run from any state $q \in Q$ to itself over the pair of words $(\varepsilon, \varepsilon)$. A run ρ from p to q is *initial* if $p \in Q_i$ and *final* if $q \in Q_f$. A run that is both initial and final is an *accepting* run of $\mathcal{T}$.

A transducer $\mathcal{T}$ defines a relation, denoted $R_\mathcal{T}$, as a subset of $\Sigma^* \times \Sigma^*$ where $(u, v) \in R_\mathcal{T}$ if and only if there exists an accepting run ρ over (u, v). The domain $\mathrm{dom}(R_\mathcal{T})$ of $R_\mathcal{T}$ is a subset of Σ^* defined as $\{u \mid (u, v) \in R_\mathcal{T}\}$. An NFT $\mathcal{T}$ is *length-preserving* if for all (u, v) in $R_\mathcal{T}$, $|u| = |v|$. Given an integer $k \geqslant 0$, an NFT $\mathcal{T}$ is said to be k-*bounded* if $d(u, v) \leqslant k$ for all $(u, v) \in R_\mathcal{T}$. It is *bounded* if it is k-bounded for some $k \geqslant 0$. Two NFTs $\mathcal{T}$ and $\mathcal{T}'$ are said to be *equivalent* if $R_\mathcal{T} = R_{\mathcal{T}'}$. Throughout this article, we assume that all NFTs are trimmed. Note that automata and hence transducers can be made trim in LogSpace using repeated calls to the NL-complete Directed Graph Reachability Problem [15].

In this article, we consider two metrics for the size of a transducer. When possible we only consider the number of states, denoted $|Q|$. However, as we consider non-deterministic transducers that can read and output finite but arbitrarily long words, we sometime refer to the size of $|\mathcal{T}|$, by which we mean the total size needed to represent $\mathcal{T}$ on the tape of a Turing Machine. In particular, it takes into account Q but also all transitions, written as quadruplets where the indexes of the states are written in binary and the input and output words are written as such.

Finally, we also introduce two functions that will be used for some proofs. Given an NFT $\mathcal{T}$, and a run ρ of $\mathcal{T}$ over some (u, v):

- We define the function in_ρ from $[1, |u|]$ to $[1, |\rho|]$ associating an index i in u to the index of the transition reading the i-th letter of the input u_i in ρ.
- Similarly, we define the function out_ρ from $[1, |v|]$ to $[1, |\rho|]$ which associates an index j of v to the index of the transition writing the j-th letter of the output v_j in ρ.

When the run ρ is clear from context, we simply write $\mathrm{in}(i)$ and $\mathrm{out}(j)$.

Shift of a run. In this contribution, the *shift* of a run is the difference in length between what is read and what is produced. Formally, the shift of a transition $\delta = (p, u, v, q)$ of an NFT is denoted by $\mathrm{s}(\delta)$ and is equal to $|u| - |v|$. The *maximum transition shift* of an NFT $\mathcal{T}$ is denoted by $\mathrm{s}_{\max}(\mathcal{T})$, and it is equal to $\max_{\delta\in\Delta}(|\,\mathrm{s}(\delta)|)$. The notion of shift extends to runs as follows: given a run ρ of an NFT over (u, v), the *shift of the run*, also denoted by $\mathrm{s}(\rho)$, is equal to $|u| - |v|$.

▶ Remark 6. Given a non-empty run $\rho = \delta_1\delta_2\cdots\delta_n$ of an NFT $\mathcal{T}$ we have $\mathrm{s}(\rho) = \sum_{i=1}^{n} \mathrm{s}(\delta_i)$.

In parallel, the *length* of a transition $\delta = (p, u, v, q)$ of an NFT is denoted $|\delta|$ and is equal to $|u| + |v|$. The *maximum transition length* of an NFT $\mathcal{T}$ is denoted $\ell_{\max}(\mathcal{T})$ and is equal to $\max_{\delta\in\Delta}(|\delta|)$.

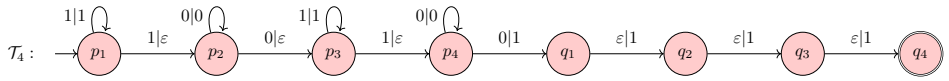

Figure 1 A bounded NFT with 8 states, 11 atomic transitions, creating at most 10 mismatches.

▶ **Example 7.** We give an example of an NFT $\mathcal{T}_4$ in Figure 1. It realizes the relation:

$$R_{\mathcal{T}_4} = \{(1^{n_1+1}0^{n_2+1}1^{n_3+1}0^{n_4+1}, 1^{n_1}0^{n_2}1^{n_3}0^{n_4}1^4) \mid n_1, n_2, n_3, n_4 \in \mathbb{N}\}.$$

The pair $(1001110000, 0110001111)$ is in $R_{\mathcal{T}_4}$ and produces $\sum_{i=1}^{4} i = 10$ mismatches. We now generalize this construction.

▶ **Lemma 8.** *There exists a family of NFTs $(\mathcal{T}_n)_{n\geqslant 2}$ such that for every $n \geqslant 2$ the NFT $\mathcal{T}_n$ has $2n$ states and satisfies $\mathsf{dev}(R_{\mathcal{T}}) \geqslant \frac{n^2+n}{2}$. Moreover, the NFT $\mathcal{T}_n$ has $3n-1$ transitions and $\mathrm{s}_{\max}(\mathcal{T}_n) = 1$.*

Proof. For each $n \geqslant 2$, we define an NFT $\mathcal{T}_n = (\{p_1, \dots, p_n\} \cup \{q_1, \dots, q_n\}, \{0,1\}, p_1, q_n, \Delta_n)$, where Δ_n is defined as follows:

$$\begin{aligned}\Delta_n = \quad & \{(p_i, i \bmod 2, \varepsilon, p_{i+1}), (p_i, i \bmod 2, i \bmod 2, p_i) \mid 1 \leqslant i < n\} \\ & \cup \{(p_n, (n+1 \bmod 2), n \bmod 2, q_1)\} \\ & \cup \{(q_i, \varepsilon, n \bmod 2, q_{i+1}) \mid 1 \leqslant i < n\}.\end{aligned}$$

The transducer $\mathcal{T}_n$ has $2n$ states and $3n-1$ transitions. The maximal shift on any given transition is 1, and it recognizes the relation $R_{\mathcal{T}_n}$ equal to

$$\{(1^{k_1+1}0^{k_2+1}\cdots(n \bmod 2)^{k_n+1}, 1^{k_1}0^{k_2}\cdots(n \bmod 2)^{k_{n-1}}(n+1 \bmod 2)^n) \mid k_1, \dots, k_n \in \mathbb{N}\}.$$

The claimed bound $\mathsf{dev}(R_{\mathcal{T}_n}) \geqslant \frac{n^2+n}{2}$ is witnessed by the pair

$$(10^21^3\cdots(n-1 \bmod 2)^{n-1}(n \bmod 2)^n, 01^20^3\cdots(n \bmod 2)^{n-1}(n+1 \bmod 2)^n) \in R_{\mathcal{T}_n}$$

which occurs by setting $k_i = i-1$ for every $1 \leqslant i \leqslant n-1$. ◀

3 Equivalence of the Comparison and Deviation Problems

This section is devoted to the proof of Theorem 1, i.e. the two-way logspace many-one reductions between deviation and comparison problems. More precisely, we show two reductions that preserve the distance and hence the exact bound. As such, the same reductions can be applied to show equivalence of the three pairs of problems.

▶ **Proposition 9.** *For all pairs of NFT $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\mathrm{dom}(\mathcal{T}_1) = \mathrm{dom}(\mathcal{T}_2)$, there exists an NFT $\mathcal{T}$ such that $(u,v) \in R_{\mathcal{T}}$ if and only if there exists an input word x satisfying $(x,u) \in R_{\mathcal{T}_1}$ and $(x,v) \in R_{\mathcal{T}_2}$. Furthermore, $\mathcal{T}$ is computable in LogSpace.*

Proof. To achieve the reduction we first transform both NFTs into input-atomic transducers, i.e. each transition either read a letter or ε. This construction can be done in LogSpace. Next we add $(\varepsilon, \varepsilon)$-transitions from each state to itself. After those two transformations, we compute the composition of the two resulting NFTs.

Let $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$, and $\mathcal{S} = (P, \Sigma, P_i, P_f, \Theta)$ be two NFTs such that $\mathrm{dom}(\mathcal{T}) = \mathrm{dom}(\mathcal{S})$. We transform $\mathcal{T}$ and $\mathcal{S}$ into $\mathcal{T}'$ and $\mathcal{S}'$ respectively, two input-atomic transducers such that $R_{\mathcal{T}'} = R_{\mathcal{T}}$, and $R_{\mathcal{S}'} = R_{\mathcal{S}}$. We also add $(\varepsilon, \varepsilon)$-transitions to all states in both

$\mathcal{S}'$ and $\mathcal{T}'$: $\Theta' = \Theta' \cup \{(q, \varepsilon, \varepsilon, q) \mid q \in P'\}$, and $\Delta' = \Delta \cup \{(q, \varepsilon, \varepsilon, q) \mid q \in Q'\}$. Since they are atomic, these constructions are also in LogSpace. Finally, we construct the NFT $\mathcal{Z} = (O, \Sigma, O_i, O_f, \Lambda)$ such that $O = Q' \times P'$, $O_i = Q'_i \times P'_i$, $O_f = Q'_f \times P'_f$, and $\Lambda \subseteq O \times \Sigma^* \times \Sigma^* \times O$, with $\Lambda = \{((q,p), u, v, (q',p')) \mid \exists x\ (q, x, u, q') \in \Delta'' \wedge (p, x, v, p') \in \Theta'\}$. By construction, we have that $(u, v) \in R_{\mathcal{Z}}$ if and only if $(x, u) \in R_{\mathcal{T}}$, and $(x, v) \in R_{\mathcal{S}}$.

Constructing the set of states can be done by storing the indexes of both states, hence it is in LogSpace, and constructing the set of transitions can be done by storing the indexes of states and the value $x \in \Sigma \cup \{\varepsilon\}$ and reading and writing both transitions. The composition of this transformation with the atomization of the transducers can be done in LogSpace in the following way: the algorithm simulates the second machine and the position of the reading head of the simulation in LogSpace. Each time the simulation asks for the k-th bit of its input, the algorithm launches a LogSpace simulation of the first machine up to its k-th production. The complete construction is therefore in LogSpace. ◀

▶ **Proposition 10.** *For all NFT $\mathcal{T}$, there exists two NFTs $\mathcal{T}_1$ and $\mathcal{T}_2$ computable in $O(1)$-space such that $\mathrm{dom}(\mathcal{T}_1) = \mathrm{dom}(\mathcal{T}_2)$, and $(u, v) \in R_{\mathcal{T}}$ if and only if there exists an input word x satisfying $(x, u) \in R_{\mathcal{T}_1}$, $(x, v) \in R_{\mathcal{T}_2}$.*

Proof. Given an NFT $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$, we let $\mathcal{T}_2 = \mathcal{T}$, and we construct $\mathcal{T}_1 = (Q, \Sigma, Q_i, Q_f, \Delta')$, where $\Delta' = \{(q, u, u, p) \mid (q, u, v, p) \in \Delta\}$. Then $R_{\mathcal{T}_1}$ is the identity function restricted to the domain of $R_{\mathcal{T}}$, hence $(u, v) \in R_{\mathcal{T}}$ if and only if there exists x, namely $x = u$, such that $(x, u) \in R_{\mathcal{T}_1}$ and $(x, v) \in R_{\mathcal{T}_2}$. Constructing the NFTs $\mathcal{T}_i$ amounts to read and copy it twice, which can be done by a two-way transducer, and hence in constant space. ◀

4 Lower Bounds

This section is devoted to proving the lower bounds stated in Theorems 2-4. We establish the results separately for each complexity class, by reductions from standard complete problems for NL (Subsection 4.1), co-NP (Subsection 4.2), and DP (Subsection 4.3), respectively.

4.1 NL-hardness

We recall the following NL-complete problem [15, Theorem 8.4].

Directed Reachability Problem
Input: A directed graph $G = (V, E)$ and two vertices $s, t \in V$.
Output: True if there exists a path from s to t in G, False otherwise.

Note that the following reductions for the NL class produce trim transducers without $(\varepsilon, \varepsilon)$-transitions. This shows that the hardness is intrinsic to the problems, and does not arise from auxiliary tasks such as eliminating ε-transitions or trimming the transducer.

In both reductions, we reduce the directed reachability problem by embedding the input graph into the transition structure of a transducer, adding an initial state and a final state connected to all vertices to ensure that the resulting transducer is trim. We then add a few transitions that introduce mismatches when a specific path exists. This actually results in a reduction from the *complement* of the reachability problem, which is equivalent since, by the Immerman–Szelepcsényi Theorem, NL = co-NL [11, 23].

▶ **Proposition 11.** *The Bounded Deviation Problem is NL-hard.*

Proof. Let $G = (V, E)$ and $s, t \in V$ be an instance of the Directed Reachability Problem. We construct a trim NFT $\mathcal{T} = (V \cup \{q_i, q_f\}, \{a, b\}, \{q_i\}, \{q_f\}, \Delta)$, where Δ is defined as follows:

$$\Delta = \{(q_i, a, a, v), (v, a, a, q_f) \mid v \in V\} \cup \{(u, a, a, v) \mid (u, v) \in E\} \cup \{(t, a, b, s)\}.$$

This construction is realizable in LogSpace. Note that all the transitions read and output the letter a, except the transition (t, a, b, s), which introduces a mismatch. Assume first that there is a path from s to t in G. This path induces a corresponding run in $\mathcal{T}$ from s to t, which forms a cycle once concatenated with transition (t, a, b, s). Iterating this cycle arbitrarily many times yields runs with arbitrarily many mismatches. As a consequence, $\mathsf{dev}(R_\mathcal{T}) = \infty$. Conversely, assume there is no path from s to t in G. Then the transition (t, a, b, s) occurs at most once in each run of $\mathcal{T}$, thus $\mathsf{dev}(R_\mathcal{T}) \leqslant 1$. Therefore, $\mathsf{dev}(R_\mathcal{T}) < \infty$ if and only if t is not reachable from s in G, which concludes the proof. ◀

▶ **Proposition 12.** *The Threshold-bounded Deviation Problem is NL-hard for all fixed $k \geqslant 1$.*

Proof. Let $G = (V, E)$ and $s, t \in V$ be an instance of the Directed Reachability Problem. We construct a trim NFT $\mathcal{T} = (V \cup \{q_i, q_f\}, \{a, b\}, \{q_i\}, \{q_f\}, \Delta)$, where Δ is defined as follows:

$$\Delta = \{(q_i, a, a, v), (v, a, a, q_f) \mid v \in V\} \cup \{(u, a, a, v) \mid (u, v) \in E\} \cup \{(q_i, a^k, b^k, s), (t, a, b, q_f)\}.$$

All the transitions read and output the letter a, except the transitions (q_i, a^k, b^k, s) and (t, a, b, q_f), which introduce k and 1 mismatches respectively. If t is reachable from s in G, there exists a run of $\mathcal{T}$ that starts with (q_i, a^k, b^k, s), then follows a path of G from s to t, and concludes with the transition (t, a, b, q_f). This run contains exactly $k + 1$ mismatches, hence $\mathsf{dev}(R_\mathcal{T}) > k$. Conversely, if t is not reachable from s, then no run can contain both (q_i, a^k, b^k, s) and (t, a, b, q_f). Hence, every run contains at most k mismatches, thus $\mathsf{dev}(\mathcal{T}) \leqslant k$. Therefore, $\mathsf{dev}(R_\mathcal{T}) \leqslant k$ if and only if t is not reachable from s in G, which concludes the proof. ◀

4.2 co-NP hardness

We recall the following NP-complete problem [15, Theorem 6.1].

3-SAT
Input: A Boolean formula $\varphi = \bigwedge_i c_i$ where each clause c_i is a disjunction of three literals.
Output: True if φ is satisfiable, False otherwise.

▶ **Proposition 13.** *The Threshold-bounded Deviation Problem is co-NP-hard.*

Proof. Let φ be a 3-SAT instance on n variables composed of m clauses:

$$\varphi(x_1, \ldots, x_n) = \bigwedge_{i=1}^{m} \left(\ell_i^1 \vee \ell_i^2 \vee \ell_i^3\right), \text{ where } \ell_i^j \text{ is either } x_k \text{ or } \neg x_k, \text{ with } 1 \leqslant k \leqslant n.$$

We construct an NFT $\mathcal{T}$ such that $\mathsf{dev}(R_\mathcal{T}) > n \cdot (m + 1) - 1$, if and only if the 3-SAT instance has a solution. The construction of $\mathcal{T}$ is based on a sequential concatenation of gadgets. Intuitively, for each clause c_i we construct an NFT $\mathcal{T}_i$ that reads a word $u \in \{0, 1\}^n$ encoding a valuation, accepts if the valuation satisfies the clause, and produces its bitwise negation. To evaluate all the clauses, these gadgets are then sequentially combined with an initial shift of minus n. The proof of correctness then relies on the fact that the input of a gadget produces n mismatches with the output of its predecessor if and only if they both read the same valuation. An example of a clause gadget can be found in Figure 2. The different gadgets and their combination is given in Figure 3 for two clauses and four variables.

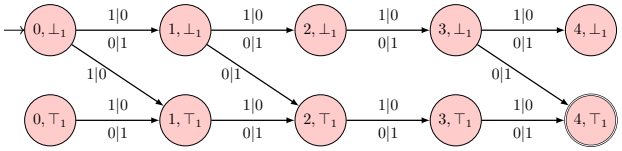

Figure 2 Gadget for $(x_1 \vee \neg x_2 \vee \neg x_4)$.

Clause gadgets. For each clause c_i, the NFT $\mathcal{T}_i$ is defined as $(Q_i, \{0,1\}, \{s_i\}, \{f_i\}, \Delta_i)$, where $Q_i = [0,n] \times \{\top_i, \bot_i\}$, $s_i = (0, \bot_i)$, $f_i = (n, \top_i)$, and Δ_i contains the following quadruplets:

1. For all $j < n$ and $b \in \{0,1\}$, $\big((j, \top_i), b, 1-b, (j+1, \top_i)\big) \in \Delta_i$.
2. For all $j < n$ and $b \in \{0,1\}$, $\big((j, \bot_i), b, 1-b, (j+1, \bot_i)\big) \in \Delta_i$.
3. For all k such that the literal x_k occurs in c_i, $\big((k-1, \bot_i), 1, 0, (k, \top_i)\big) \in \Delta_i$.
4. For all k such that the literal $\neg x_k$ occurs in c_i, $\big((k-1, \bot_i), 0, 1, (k, \top_i)\big) \in \Delta_i$.

Note that every accepting run of $\mathcal{T}_i$ is of length exactly n, and outputs the negation of its input. Furthermore, the transitions of type 3 and 4 above are the only ones that move from the $\bot_i$-states to the $\top_i$-states, and thus are necessary to reach the final state. They are enabled exactly when the corresponding literal makes the clause c_i true, which ensures that the accepting runs of $\mathcal{T}_i$ encode valuations satisfying c_i.

Boundary gadgets. We construct two gadgets $\mathcal{T}_{\text{init}} = (Q_{\text{init}}, \{0,1\}, \{s_{\text{init}}\}, \{f_{\text{init}}\}, \Delta_{\text{init}})$ and $\mathcal{T}_{\text{final}} = (Q_{\text{final}}, \{0,1\}, \{s_{\text{final}}\}, \{f_{\text{final}}\}, \Delta_{\text{final}})$, where:

- $Q_{\text{init}} = [0,n] \times \{\text{init}\}$, $Q_{\text{final}} = [0,n] \times \{\text{final}\}$,
- $s_{\text{init}} = (0, \text{init})$, $s_{\text{final}} = (0, \text{final})$,
- $f_{\text{init}} = (n, \text{init})$, $f_{\text{final}} = (n, \text{final})$,
- $\Delta_{\text{init}} = \{\big((i, \text{init}), \varepsilon, b, (i+1, \text{init})\big) \mid 0 \leqslant i < n \text{ and } b \in \{0,1\}\}$,
  $\Delta_{\text{final}} = \{\big((i, \text{final}), b, \varepsilon, (i+1, \text{final})\big) \mid 0 \leqslant i < n \text{ and } b \in \{0,1\}\}$.

Note that $R_{\mathcal{T}_{\text{init}}} = \{\varepsilon\} \times \{0,1\}^n$ and $R_{\mathcal{T}_{\text{final}}} = \{0,1\}^n \times \{\varepsilon\}$.

Global construction. We construct an NFT $\mathcal{T}$ that recognizes the concatenation of the relations of all the gadgets: $R_{\mathcal{T}} = R_{\mathcal{T}_{\text{init}}} \cdot R_{\mathcal{T}_1} \cdot \ldots \cdot R_{\mathcal{T}_m} \cdot R_{\mathcal{T}_{\text{final}}}$. Since each gadget has a unique initial state with no incoming transition and a unique final state with no outgoing transition, this concatenation is achieved by merging the final state of each gadget with the initial state of the following one : f_{init} with s_1, f_i with s_{i+1} for all $1 \leqslant i < m$, and f_m with s_{final}. The initial state of $\mathcal{T}$ is s_{init} and the final state is f_{final}. The construction is then polynomial, the number of states of $\mathcal{T}$ being $(n+1) \cdot (2m+2) - (m+1) = (2n+1) \cdot (m+1)$.

Correctness. We prove that φ is satisfiable if and only if there exists (y,z) in $R_{\mathcal{T}}$ such that $d(y,z) \geqslant n \cdot (m+1)$. Suppose that φ is satisfiable and let ν be a valuation satisfying it. Let $u = \nu(x_1) \cdots \nu(x_n)$ be the word of length n encoding this valuation and let $v = \neg\nu(x_1) \cdots \neg\nu(x_n)$ be its negation. Since ν is a valuation satisfying φ, it satisfies each of its clauses, hence for all $i \in [1,m]$, $(u,v) \in R_{\mathcal{T}_i}$. Furthermore, $(\varepsilon, v) \in R_{\mathcal{T}_{\text{init}}}$ and $(u, \varepsilon) \in R_{\mathcal{T}_{\text{final}}}$. Consequently, $(u^{m+1}, v^{m+1}) \in R_{\mathcal{T}}$, with $d(u^{m+1}, v^{m+1}) = |u| \cdot (m+1) = n \cdot (m+1)$.

Now let $(y,z) \in R_{\mathcal{T}}$, since $R_{\mathcal{T}} = R_{\mathcal{T}_{\text{init}}} \cdot R_{\mathcal{T}_1} \cdot \ldots \cdot R_{\mathcal{T}_m} \cdot R_{\mathcal{T}_{\text{final}}}$, we can decompose y and z into $u_1 \cdots u_m u_{\text{final}}$ and $v_{\text{init}} v_1 \cdots v_m$ respectively, with for all $i \in [1,m]$, $|v_{\text{init}}| = |u_{\text{final}}| = |u_i| = |v_i| = n$. By definition of $R_{\mathcal{T}_i}$, u_i describes a valuation satisfying c_i and v_i is the negation of u_i. If $d(y,z) \geqslant n \cdot (m+1)$ while $|y| = |z| = n \cdot (m+1)$, all positions of y and z mismatch. Consequently, u_1 is the negation of v_{init}, for all $i \in [2,m]$ u_i is the negation of v_{i-1}, and u_{final} is the negation of v_m. Therefore, $u_1 = u_2 = \ldots = u_m = u_{\text{final}}$, and they all encode the same valuation satisfying $c_1, c_2, \ldots, c_m$ and consequently satisfying φ. ◀

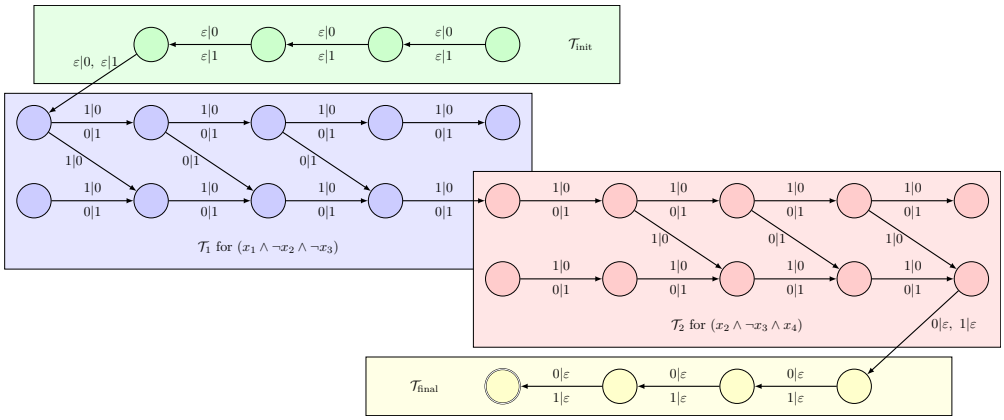

Figure 3 An NFT $\mathcal{T}$ for a SAT formula with 2 clauses and 4 variables.

4.3 DP-Hardness

A canonical example of a DP problem is SAT-UNSAT [16, S. 2, Lemma 1]:

SAT-UNSAT Problem
Input: Two Boolean formulas φ_1, φ_2.
Output: TRUE if φ_1 is satisfiable and φ_2 is not satisfiable, FALSE otherwise.

▶ **Proposition 14.** *The Exact Deviation Problem is DP-hard.*

Proof. Consider an instance of SAT-UNSAT given by two propositional formulas φ_1 and φ_2. We reuse the construction of the proof of the Proposition 13 to obtain two NFTs $\mathcal{T}_1$, $\mathcal{T}_2$ and two positive integers k_1, k_2 such that for $i \in \{1, 2\}$, $|u| = |v| = k_i$ for all $(u, v) \in R_{\mathcal{T}_i}$, and:

$$\begin{cases} \mathsf{dev}(R_{\mathcal{T}_i}) \leqslant k_i - 1 & \text{if } \varphi_i \text{ is not satisfiable,} \\ \mathsf{dev}(R_{\mathcal{T}_i}) = k_i & \text{if } \varphi_i \text{ is satisfiable.} \end{cases}$$

We then create $\mathcal{T}_1'$ by concatenating k_2 copies of $\mathcal{T}_1$ together, i.e., $R_{\mathcal{T}_1'} = R_{\mathcal{T}_1}^{k_2}$. Therefore:

$$\begin{cases} \mathsf{dev}(R_{\mathcal{T}_1'}) \leqslant (k_1 - 1)k_2 & \text{if } \varphi_1 \text{ is not satisfiable,} \\ \mathsf{dev}(R_{\mathcal{T}_1'}) = k_1 k_2 & \text{if } \varphi_1 \text{ is satisfiable.} \end{cases}$$

Consider $\mathcal{T}$ the concatenation of $\mathcal{T}_1'$ and $\mathcal{T}_2$. By construction, for all $(u, v) \in R_{\mathcal{T}}$, $|u| = |v| = k_1 k_2 + k_2$ and $\mathcal{T}$ is a bounded NFT. Furthermore, φ_1 and φ_2 are both satisfiable if and only if there exists $(u, v) \in R_{\mathcal{T}}$ such that $d(u, v) = k_1 k_2 + k_2$. Note also that if φ_1 is not satisfiable, for all $(u, v) \in R_{\mathcal{T}_1'}$, there will be at most $(k_1 - 1)k_2$ mismatches from $\mathcal{T}_1'$ and at most k_2 mismatches from $\mathcal{T}_2$, there will be at most $k_1 k_2$ mismatches. The SAT-UNSAT instance is true if and only if the bound of $\mathcal{T}$ is greater than $k_1 k_2$, and strictly less than $k_1 k_2 + k_2$.

We conclude by reducing this interval to a single value in the following way. Let $\mathcal{T}_3$ be an NFT with two states and one transition such that $R_{\mathcal{T}_3} = \{(0^{k_2 - 1}, 1^{k_2 - 1})\}$. Consider $\mathcal{S}$ the NFT such that $\mathcal{S}$ is the concatenation of $\mathcal{T}_1'$ with the non-deterministic choice between $\mathcal{T}_2$ and $\mathcal{T}_3$. We have $R_{\mathcal{S}} = R_{\mathcal{T}_1'} \cdot (R_{\mathcal{T}_2} \cup R_{\mathcal{T}_3})$. In the end, the SAT-UNSAT instance is true if and only if the bound of $\mathcal{S}$ is exactly $k_1 k_2 + k_2 - 1$. ◀

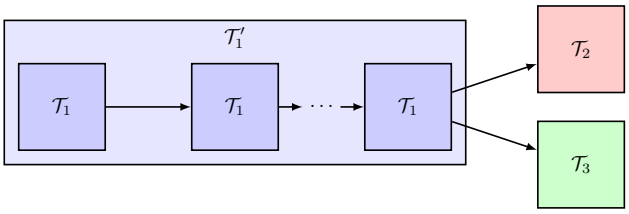

Figure 4 Resulting NFT $\mathcal{S}$ for SAT-UNSAT to Exact Deviation Problem reduction.

5 Upper Bounds

This section is devoted to proving the upper bounds stated in Theorems 2 and 3. In Subsection 5.1 we show that the Bounded Deviation Problem is in NL and that whenever an NFT $\mathcal{T}$ is bounded, the bound is at most quadratic in $|\mathcal{T}|$ (Proposition 15). Then, Subsection 5.2 proves the upper bound for the Threshold-bounded Deviation Problem. In the following section, we construct indiscriminately NL or co-NL algorithms, as they are proven to be equivalent [11, 23].

5.1 Bounded Deviation Problem

This subsection is dedicated to prove the following upper bound:

▶ **Proposition 15.** *The Bounded Deviation Problem is in NL. Moreover, if $\mathcal{T}$ is bounded, then its deviation is in $O(|\mathcal{T}|^2)$.*

We prove Proposition 15 in two steps. First, we show in Lemma 18 that bounded NFTs are characterized by a conjunction of two NL-testable properties: being length-preserving, and having, for each cyclic run, input and output words that are conjugate with respect to a state-dependent shift.

To formalize the notion of state-dependent shift used in the loop conjugacy property, we first establish the following lemma. It shows that, in a length-preserving NFT, all runs leading to a given state p share the same shift s_p, which we call the *shift of the state p*.

▶ **Lemma 16.** *Let $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$ be a length-preserving NFT. For all states $p \in Q$, there exists $s_p \in \mathbb{Z}$ such that every run $\rho \in \Delta^*$ from $q \in Q_i$ to p satisfies $\mathrm{s}(\rho) = s_p$. Moreover, $|s_p| \leqslant \min(\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|, |\mathcal{T}|)$.*

▶ Remark 17. We discuss here the double bound on s_p. These bounds are in general incomparable, and each can be advantageous depending on the transducer $\mathcal{T}$. If $\mathcal{T}$ has many transitions compared to its number of states, it is more efficient to rely on the $\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$. On the other hand, if $\mathcal{T}$ has few transitions but large shift values, then $\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$ might be close to $|\mathcal{T}|^2$, in which case the bound $|\mathcal{T}|$ is preferable. Notably, we rely on this later bound to obtain a generic quadratic bound on the deviation of $\mathcal{T}$ in Lemma 19.

We now formally express the conditions for an NFT to be bounded.

▶ **Lemma 18.** *Let $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$ be an NFT. Then $\mathcal{T}$ is bounded if and only if both:*
1. *$\mathcal{T}$ is length-preserving, and*
2. *for every run $\rho \in \Delta^*$ from $p \in Q$ to itself over some (u, v), u is conjugate to v by s_p.*

Moreover, these conditions imply that the deviation of $\mathcal{T}$ is in $O(|\mathcal{T}|^2)$ and each can be decided in NL.

▶ **Lemma 19.** *Let $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$ be an NFT and $b = \min(\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|, |\mathcal{T}|)$. If $\mathcal{T}$ is not bounded by $B = (b + \ell_{\max}(\mathcal{T}) + 2) \cdot |Q|$. Then either:*

1. *$\mathcal{T}$ is not a length-preserving NFT, or*
2. *$\mathcal{T}$ is length-preserving, but there exists $\rho \in \Delta^*$, a run from $p \in Q$ to itself over some $(u, v) \in \Sigma^* \times \Sigma^*$ such that u is not conjugate to v by s_p.*

Proof. Let $\mathcal{T}$ be an NFT not bounded by B. If it is also not length-preserving, then the lemma holds trivially. Suppose then that it is length-preserving. Let $\rho = \delta_1\delta_2 \cdots \delta_n \in \Delta^*$ be an accepting run of $\mathcal{T}$ over some (u, v) such that $d(u, v) \geqslant B$. We define $m_1, \ldots, m_{d(u,v)}$ to be the strictly increasing sequence of positions such that $u_{m_i} \neq v_{m_i}$ and $q_i = \pi_1(\mathrm{in}_\rho(m_i))$ the sequence of starting states of the transitions reading the mismatches. Since $d(u, v) \geqslant B$, there exists a state $p \in Q$ that appears at least $b + \ell_{\max}(\mathcal{T}) + 2$ times in the sequence $(q_i)_i$. We denote by o the smallest index of an occurrence of p and by r the $(\ell_{\max}(\mathcal{T}) + 1)$-th largest index of an occurrence of p. Consequently, the factor $\rho' = \delta_o \cdots \delta_r$ of ρ is a run from p to itself over some (u', v') such that at least $b+1$ mismatches are read in u'. Since s_p is bounded by b by Lemma 16 and the subrun ρ' generates more than b mismatches, at least one of them is produced by ρ'. As the initial shift is s_p, there exist two integers $i \leqslant |u'|$ and $j \leqslant |v'|$ such that $j - i \equiv s_p \bmod |u'|$ and $u'_i \neq v'_j$. Consequently, u' is not conjugate to v' by s_p. ◀

Before defining the algorithms, we establish two technical lemmas that bound the maximal deviation inside runs and the length of runs in terms of states and endpoint shifts.

▶ **Lemma 20.** *Let $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$ be a length-preserving transducer. Then for all runs $\rho \in \Delta^*$, $\mathrm{s}_{\max}(\rho) \leqslant \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$.*

Proof. Let $\rho \in \Delta^*$ be a run of a length-preserving NFT $\mathcal{T}$. By Lemma 18, cycles do not act on the shift of a run. Then there exists $\rho' \in \Delta^*$ a run of $\mathcal{T}$ without cycle such that $\mathrm{s}_{\max}(\rho) = \mathrm{s}_{\max}(\rho')$. Being without cycle, $|\rho'| \leqslant |Q|$ and by definition $\mathrm{s}_{\max}(\rho') \leqslant |\rho'| \cdot \mathrm{s}_{\max}(\mathcal{T}) \leqslant \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$. ◀

▶ **Lemma 21.** *Let $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$ be a length-preserving transducer, and $\rho \in \Delta^*$ be a run of $\mathcal{T}$ from a state $q \in Q$ to a state $p \in Q$ over some (u, v) without $(\varepsilon, \varepsilon)$-cycles. Then*

$$|\rho| \leqslant |Q| \cdot |uv| = |Q| \cdot (s_q + 2 \cdot |u| - s_p) = |Q| \cdot (s_p + 2 \cdot |v| - s_q).$$

Proof. Let $\rho \in \Delta^*$ be a run of $\mathcal{T}$ from a state $q \in Q$ to a state $p \in Q$ over some (u, v). Since there is no $(\varepsilon, \varepsilon)$-cycles, each sequence of $|Q|$ consecutive transitions either reads or writes at least one letter, hence $|\rho| \leqslant |Q| \cdot |uv|$. Then by definition of the shift and the shift of a state, we have that $|u| - |v| = s_p - s_q$. We obtain directly that $|Q| \cdot |uv| = |Q| \cdot (s_q + 2 \cdot |u| - s_p) = |Q| \cdot (s_p + 2 \cdot |v| - s_q)$. ◀

▶ **Lemma 22.** *Given an NFT $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$, it can be decided in NL whether every accepting runs $\rho \in \Delta^*$ over some (u, v) satisfies $|\rho| \leqslant |Q|$ and $|u| = |v|$.*

Proof. We construct a co-NL algorithm that accepts runs $\rho \in \Delta^*$ over $(u, v) \in (\Sigma^* \times \Sigma^*)$ such that $|\rho| \leqslant |Q|$ and $|u| \neq |v|$. The algorithm works as follows. It guesses a run of $\mathcal{T}$ transition by transition and each time, updates two integers stored in binary:

- ℓ stores the number of transitions that have already been through.
- s stores the shift between the two words already read and written.

If a final state is reached and s is not equal to 0 then the algorithm stops and accepts. If ℓ become greater than $|Q|$ then the algorithm stops and rejects. If neither of the above conditions are met, then the algorithm guesses the next transition, if it is not possible, then it stops and rejects. Denote that because we have been through at most $|Q|$ transitions, s stays in the range $[-\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|, \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|]$. Thus, this algorithm is NL. ◀

▶ **Lemma 23.** *Given an NFT $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$, it can be decided in NL whether every run $\rho \in \Delta^*$ from some $p \in Q$ to itself over some (u, v) satisfies $|u| \neq |v|$.*

Proof. First, note that if there exists a cyclic run of length greater than $|Q|$ over some (u, v) such that $|u| \neq |v|$, decomposing it into simple cycles will yield at least one cyclic subrun of length smaller than or equal to $|Q|$ over some (u', v') such that $|u'| \neq |v'|$. We construct a co-NL algorithm that accepts runs $\rho \in \Delta^*$ with $|\rho| \leqslant |Q|$ from $p \in Q$ to itself, over some (u, v) such that $|u| \neq |v|$. The algorithm works as follows. We guess a state p and a run ρ of $\mathcal{T}$ starting in p transition by transition. At each step it updates three integers stored in binary:

- p stores the first state of ρ.
- ℓ stores the number of transitions that have already been processed.
- s stores the shift between the processed input and output.

If the state p is reached and s is not equal to 0 then the algorithm stops and accepts. If ℓ become greater than $|Q|$ then the algorithm stops and rejects. If neither of the above conditions are met, then the algorithm guesses the next transition if possible, and rejects otherwise. Note that since the algorithm processes at most $|Q|$ transitions, s stays in the range $[-\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|, +\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|]$. Thus, this algorithm is NL. ◀

5.2 Threshold-bounded Deviation Problem

Now we address the Threshold-bounded Deviation Problem, and show that its complexity depends on whether the threshold k is part of the input or fixed.

▶ **Proposition 24.** *The Threshold-bounded Deviation Problem is:*

- *in co-NP when k is part of the input;*
- *in NL for every fixed $k \in \mathbb{N}$.*

Since a NFT $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$ can be bounded only if it is length-preserving, then $\mathcal{T}$ is k-bounded if and only if for all accepting runs $\rho \in \Delta^*$ over some (u, v) there is at most k mismatch between u and v. We call a run with k mismatch a *k-mismatch witness* for $\mathcal{T}$. We show that if a k-mismatch witness exists, there exists one of polynomial length on the size of $\mathcal{T}$ and consequently, we can construct an NP algorithm to find such witness. More precisely, the algorithm first checks whether the NFT is bounded, which can be done in NL thanks to Theorem 2. Then if k is greater than the quadratic bound of Lemma 19 the algorithm accepts, otherwise it applies the co-NP algorithm of the Lemma below.

▶ **Lemma 25.** *Given a bounded NFT $\mathcal{T} = (Q, \Sigma, Q_i, Q_f, \Delta)$, and an integer $k < B$ where $B = (\min(\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|, |\mathcal{T}|) + \ell_{\max}(\mathcal{T}) + 2) \cdot |Q|$, it can be decided in co-NP whether for all accepting runs $\rho \in \Delta^*$ over some (u, v), $d(u, v) \leqslant k$.*

Proof. We first claim a small witness property and give a co-NP algorithm relying on it. The claim is proved afterward.

▷ Claim 26. If there exists an accepting run $\rho \in \Delta^*$ over some (u, v) such that $d(u, v) > k$ then there exists an accepting run $\rho' \in \Delta^*$ over some (u', v') such that $|\rho'| \leqslant L = 8 \cdot \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|^3$ and $d(u', v') > k$.

From this claim, we construct a co-NP algorithm that accepts accepting runs $\rho \in \Delta^*$ of length $|\rho| \leqslant L$ over some (u, v) such that $d(u, v) > k$. The algorithm works as follows. It first guesses $k + 1$ positions, all these positions are stored in a set of pairs Z of the form (i, a), where i is an integer storing the guessed position, and a stores an element of $\Sigma \cup \{\varepsilon\}$. Initially, all pairs in Z are of the form (i, ε). Next, the algorithm guesses a run ρ of $\mathcal{T}$, transition by transition, and each step it updates four variables:

- ℓ, an integer stored in binary, tracking the number of processed transitions.
- ℓ_u, an integer stored in binary, tracking the length of the processed input.
- ℓ_v, an integer stored in binary, tracking the length of the processed output.
- Z, the set of pairs.

First, if the first state of the run ρ is not in Q_i then the algorithm stops and rejects. Next, at each step, the algorithm starts by comparing the input and output of the current transition denoted (u, v) with Z. If there exists $(i, a) \in Z$ such that i is in $(\ell_u, \ell_u + |u|]$, or in $(\ell_v, \ell_v + |v|]$ then the following updates are executed:

- If $a = \varepsilon$, and i is in both $(\ell_u, \ell_u + |u|]$ and $(\ell_v, \ell_v + |v|]$. Then, if $u_{i-\ell_u} \neq v_{i-\ell_v}$, the algorithm removes (i, ε) from the set, and if $u_{i-\ell_u} = v_{i-\ell_v}$, it stops and rejects.
- Else if $a = \varepsilon$, then the algorithm updates the value of a with $u_{i-\ell_u}$ or $v_{i-\ell_v}$ depending if i was in $(\ell_u, \ell_u + |u|]$, or in $(\ell_v, \ell_v + |v|]$.
- Else if $a \neq \varepsilon$ and a is not equal to $u_{i-\ell_u}$ or $v_{i-\ell_v}$, again depending if i was in $(\ell_u, \ell_u + |u|]$, or in $(\ell_v, \ell_v + |v|]$ then the algorithm removes (i, a) from the set.
- Else if $a \neq \varepsilon$ and a is equal to $u_{i-\ell_u}$ or $v_{i-\ell_v}$, again depending if i was in $(\ell_u, \ell_u + |u|]$, or in $(\ell_v, \ell_v + |v|]$ then the algorithm stops and rejects.

After these updates of Z the algorithm updates the three other variables ℓ, ℓ_u, and ℓ_v. If a final state is reached with $\ell \leqslant L$ and $|Z| = 0$ then the algorithm stops and accepts. If ℓ becomes greater than L, then the algorithm stops and rejects. If neither of this two conditions are met, then the algorithm guesses the next transition, if it is not possible, then it rejects.

Proof of Claim 26. Let $\rho \in \Delta^*$ be an accepting run of $\mathcal{T}$ over some (w, z) such that $k < d(w, z)$, and $|\rho| > L$. We prove that there is a valid sub-run ρ' of ρ that generates the same number of mismatches as ρ. We assume that ρ is $(\varepsilon, \varepsilon)$-cycle free, as such cycles cannot generate mismatches nor shift. Consequently, there exists a state $p \in Q$ which is visited $8 \cdot \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|^2$, and there exists $\mu = \delta_1\delta_2 \cdots \delta_n$ a factor of ρ from p to itself over some (u, v) with $|\mu| \geqslant 8 \cdot \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|^2$. We distinguish two cases, when $s_p \geqslant 0$ and when $s_p \leqslant 0$. Those two cases being symmetrical, we focus on the first one. Using Lemma 21, we get that $|uv| \leqslant 8 \cdot \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$. Moreover, thanks to Lemma 20, $||u| - |v|| \leqslant \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$, therefore, we get that both $|u| > \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$ and $|v| > \mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$. We decompose the run μ into five part, three runs and two transitions, as shown in Figure 5:

1. $\mu^s = \delta_1 \cdots \delta_{\mathrm{out}(s_q)-1}$, a run from p to q' over some (u', v'),
2. $\delta^o = \delta_{\mathrm{out}(s_q)}$, a run from p'' to q'' over some (u'', v''),
3. $\mu^m = \delta_{o+1} \cdots \delta_{m-1}$, a run from $p^{(3)}$ to $q^{(3)}$ over some $(u^{(3)}, v^{(3)})$,
4. $\delta^r = \delta_{\mathrm{in}(|u'|-s_q+1)}$, a run from $p^{(4)}$ to $q^{(4)}$ over some $(u^{(4)}, v^{(4)})$,
5. $\mu^f = \delta_{\mathrm{in}(|u'|-s_q+1)+1} \cdots \delta_n$, a run from $p^{(5)}$ to p over some $(u^{(5)}, v^{(5)})$.

Informally, δ^o marks the transition where s_p, the initial shift μ is caught up, and δ^r marks the symmetrical transition from which the input read is not matched within μ. Note that the output of μ^s is equal to s_q which is smaller than $\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|$. Therefore, using Lemma 21, we can suppose that μ^s is a $(\varepsilon, \varepsilon)$-cycle free run of length at most $|Q|(s_q + 2s_q) \leqslant 3\,\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|^2$. A symmetric argument holds for μ^f and gives the same bound on its length. Finally, the length of μ^m can be bounded by $|Q|$ thanks to Lemma 18 as any loop in μ^m can be cut off without affecting the number of mismatches. ◁

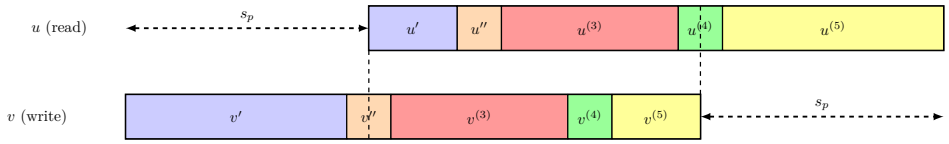

Figure 5 Illustration of the decomposition of μ into five parts.

Altogether, we proved that μ can be made to be of length smaller than

$$|\mu^s| + 1 + |\mu^m| + 1 + |\mu^f| \leqslant 6\,\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|^2 + |Q| + 1 \leqslant 8\,\mathrm{s}_{\max}(\mathcal{T}) \cdot |Q|^2.$$ ◀

6 Conclusion

In this work, we introduced some new lower-bounds on the complexity of three NFTs comparison problems for pairs of transducers. We also strengthened the already studied upper-bounds for those problems in [2]. A natural direction for pursuing this work would be to extend these positive results to other metrics over pair of words, such as the Levenshtein edit distance [12] or the Dynamic Time Warping distance (or DTW) [24, 18] which is notably used in speech recognition or DNA sequencing for the comparison of gene expression [20].

In another direction, the NL-complexity that we achieved for the Bounded Comparison Problem and more importantly the fixed Threshold Comparison Problem could be leveraged for practical use. In particular, one can think of applying these to the fixed approximated Model-Checking, which asks whether a specification can be satisfied by a class of machine up to tolerating a fixed maximal number of errors. In our setting, the specification is given by the first transducer which can be highly ambiguous, while the model is given by the second transducer which would be a deterministic one. Deciding the fixed threshold comparison amounts then to decide the fixed approximated Model-Checking. In a similar but more ambitious vein, we hope to extend this approach to approximated synthesis, by adapting our approach to directly generate, from the specification given as an ambiguous transducer, a deterministic transducer whose deviation from the specification is bounded.

References

1 C. Aiswarya, Amaldev Manuel, and Saina Sunny. Edit distance of finite state transducers. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, Tallinn, Estonia, July 8-12, 2024*, volume 297 of *LIPIcs*, pages 125:1–125:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.125.

2 C. Aiswarya, Amaldev Manuel, and Saina Sunny. Edit distance of finite state transducers. *CoRR*, abs/2404.16518, 2024. doi:10.48550/arXiv.2404.16518.

3 Christian Choffrut and Giovanni Pighizzini. Distances between languages and reflexivity of relations. *Theor. Comput. Sci.*, 286(1):117–138, 2002. doi:10.1016/S0304-3975(01)00238-9.

4 Emmanuel Filiot, Ismaël Jecker, Khushraj Madnani, and Saina Sunny. Approximate problems for finite transducers. In Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis, editors, *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, Aarhus, Denmark, July 8-11, 2025*, volume 334 of *LIPIcs*, pages 155:1–155:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.155.

5 Emmanuel Filiot and Pierre-Alain Reynier. Transducers, logic and algebra for functions of finite words. *ACM SIGLOG News*, 3(3):4–19, 2016. doi:10.1145/2984450.2984453.

6 Seymour Ginsburg and Gene F. Rose. A characterization of machine mappings. *Journal of Symbolic Logic*, 33(3):468–468, 1968. doi:10.2307/2270340.

7 R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x.

8 R. W. Hamming. *Coding and information theory*. Prentice-Hall, Englewood Cliffs, NJ, 2nd ed edition, 1986.

9 Yo-Sub Han, Sang-Ki Ko, and Kai Salomaa. Computing the edit-distance between a regular language and a context-free language. In Hsu-Chun Yen and Oscar H. Ibarra, editors, *Developments in Language Theory - 16th International Conference, DLT 2012, Taipei, Taiwan, August 14-17, 2012. Proceedings*, volume 7410 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 2012. doi:10.1007/978-3-642-31653-1_9.

10 Thomas A. Henzinger, Jan Otop, and Roopsha Samanta. Lipschitz robustness of finite-state transducers. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, New Delhi, India, December 15-17, 2014*, volume 29 of *LIPIcs*, pages 431–443. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.431.

11 Neil Immerman. Nondeterministic space is closed under complementation. In *Proceedings: Third Annual Structure in Complexity Theory Conference, Georgetown University, Washington, D. C., USA, June 14-17, 1988*, pages 112–115. IEEE Computer Society, 1988. doi:10.1109/SCT.1988.5270.

12 Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, February 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

13 Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *Int. J. Found. Comput. Sci.*, 14(6):957–982, 2003. doi:10.1142/S0129054103002114.

14 Anca Muscholl and Gabriele Puppis. The many facets of string transducers (invited talk). In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, Berlin, Germany, March 13-16, 2019*, volume 126 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.2.

15 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

16 Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *J. Comput. Syst. Sci.*, 28(2):244–259, April 1984. doi:10.1016/0022-0000(84)90068-0.

17 George N. Raney. Sequential functions. *J. ACM*, 5(2):177–180, 1958. doi:10.1145/320924.320930.

18 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pages 43–49, 1978.

19 Roopsha Samanta, Jyotirmoy V. Deshmukh, and Swarat Chaudhuri. Robustness analysis of string transducers. In Dang Van Hung and Mizuhito Ogawa, editors, *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, volume 8172 of *Lecture Notes in Computer Science*, pages 427–441. Springer, 2013. doi:10.1007/978-3-319-02444-8_30.

20 Helena Skutkova, Martin Vítek, Petr Babula, René Kizek, and Ivo Provazník. Classification of genomic signals using dynamic time warping. *BMC Bioinform.*, 14(S-10):S1, 2013. doi:10.1186/1471-2105-14-S10-S1.

21 Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong, editors, *Proceedings of the 5th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1973, Austin, Texas, USA*, pages 1–9. ACM, 1973. doi:10.1145/800125.804029.

22 Saina Sunny. *Distance and Conjugacy of Word Transducers*. Phd thesis, Indian Institute ot Technology Goa, 2025. available at `https://saisunny1994.github.io/`.

23 Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988. `doi:10.1007/BF00299636`.

24 T. K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4:52–57, 1968. Russian Kibernetika 4(1):81-88 (1968).