Regular Repair of Specifications

Michael Benedikt Gabriele Puppis Cristian Riveros

***Abstract*—What do you do if a computational object (e.g. program trace) fails a specification? An obvious approach is to perform *repair*: modify the object minimally to get something that satisfies the constraints. In this paper we study repair of temporal constraints, given as automata or temporal logic formulas. We focus on determining the number of repairs that must be applied to a word satisfying a given input constraint in order to ensure that it satisfies a given target constraint. This number may well be unbounded; one of our main contributions is to isolate the complexity of the "bounded repair problem", based on a characterization of the pairs of regular languages that admit such a repair. We consider this in the setting where the repair strategy is unconstrained and also when the strategy is restricted to use finite memory. Although the streaming setting is quite different from the general setting, we find that there are surprising connections between streaming and non-streaming, as well as within variants of the streaming problem.**

I. Introduction

When a computational object does not satisfy a specification, an obvious approach is to *repair* it – edit it minimally so that it becomes valid. We may want to perform this editing transformation on the object, or we may be merely interested in knowing how difficult it would be to perform – that is, determining how far a given object or collection of objects is from satisfying the specification. In the database community, this has been extensively studied under the notion of *constraint repair* (see e.g. [1], [2]): the specifications considered there are relational integrity constraints, such as keys and foreign keys, and the problems considered include determining how much a database needs to be modified in order to satisfy a given constraint.

Here we initiate the study of repair for temporal constraints on words. The notion of repairing a word is indeed more obvious than in the case of databases: we can simply consider the *edit distance* between strings, a standard measure of how many basic operations it takes to get from one string to the next. Edit distance is lifted in a natural way to give a measure $dist(w, L)$ of the distance of a string w to a language (collection of strings) L: the minimal distance of w to any string in L. It is well-known [3] that the standard dynamic programming approach to edit distance extends to give an efficient algorithm for calculating $dist(w, L)$ when L is a regular language given as an NFA.

In this work we take the next step and consider a "distance" between languages – given languages R and T (specified in different ways) we aim to calculate how difficult it is to transform a string satisfying R into a string satisfying T. The notation is motivated by considering R to be a *restriction* — a constraint that the input is guaranteed to satisfied – while T is a *target* – a constraint that we want to enforce. We consider the worst-case over a string $s \in R$ of the number of edit operations needed to move s into R: $sup_{w \in R} dist(w, T)$. That is, we look at the worst-case number of operations needed to get from R to T. Of course, this number may be infinite; the core of our results is a procedure for solving the *bounded repair problem* – determining whether the supremum above is finite. Of course, in order to compute this effectively, we need to restrict the languages R and T. We consider this problem for regular languages, presented by both deterministic and non-deterministic finite automata. We also consider languages specified by Linear temporal logic. In all these cases we determine the complexity of the bounded repair problem.

Above we considered the use of an edit/correction function that can read the whole string in memory. In this work we consider the impact of limitations on the editing process – what happens when we require the editing to be done by a transducer, reading in the input letter-by-letter and producing the corrected output, based only on a finite amount of control state and a fixed amount of lookahead in the word. We refer to this as a *streaming repair processor*. We isolate the complexity of the streaming repair problem, for any lookahead, for any of the language classes considered in the non-streaming case.

The above deals with determining whether the distance of one specification to another is finite or infinite. But in the finite case, we may want to compute this distance exactly, and to produce the processor that optimally edits a given specification. Note that in the non-streaming setting, it is easy to describe the optimal processor: it is simply the function that given a word w runs a dynamic-programming algorithm to compute the edit distance to the target (e.g. the algorithm from [3]) in the case of NFAs). In the streaming setting, it is not clear how to derive the optimal editing algorithm efficiently. We give results on the complexity of computing the exact bound when it is finite in both the streaming and non-streaming setting, and also give procedures for computing the optimal processor in the streaming setting.

The streaming and non-streaming repair problems have very different flavors: the former are closely related to games played on the components of an automata, while the latter require a more global analysis, and exhibit a close relation to *distance automata*. However, there are connections between the different problems: we show that in the case where there is no restriction, the bounded repair problems are the same for both the streaming and non-streaming setting. We also show that the bounded repair problem in the streaming setting is independent of the lookahead, and is robust under plausible alternative definitions.

In summary our contributions are:

- We formalize the bounded repair problem for languages and characterize when regular languages have bounded repair, in both the streaming and non-streaming setting.
- We show that the bounded repair problem in the streaming setting is independent of the lookahead of the stream processor, and is robust under variants of the cost function.
- Using the characterizations above, we give results on the complexity of the bounded repair problem in each setting.
- We present results on the complexity of computing the optimal bound, and on computing the optimal strategy in the streaming case.
- We demonstrate special cases where the streaming and non-streaming bounded repair problems have the same solution.

Organization: Section II gives preliminaries, while Section III defines the basic problems. Section IV gives the characterizations of bounded repair that we will use throughout the remainder. Section V studies the non-streaming case, while Section VI deals with the streaming case. Section VIII briefly discusses extensions to infinite words, while Section IX gives related work and conclusions. Proofs are relegated to an appendix.

II. Basic notation and terminology

Given a word w over an alphabet Σ, we denote by $|w|$ its length and, for every position $1 \leq i \leq j \leq |w|$, we denote by $w[i]$ (resp., $w[..i]$, $w[i..]$, $w[i..j]$) the i-th symbol of w (resp., the prefix of w ending at position i, the suffix of w starting at position i, the infix of w starting at position i and ending at position j).

Automata. Non-deterministic finite state automata (shortly, *NFA*) will be represented by tuples of the form $\mathcal{A} = (\Sigma, Q, E, I, F)$, where Σ is a finite alphabet, Q is a finite set of states, $E \subseteq Q \times \Sigma \times Q$ is a transition relation, and $I, F \subseteq Q$ are sets of initial and finite states. The notions of run and accepted word are the usual ones. $\mathscr{L}(\mathcal{A})$ is the language recognized by $\mathcal{A}$. For technical reasons, it is also convenient to assume that all states of an NFA are reachable from some initial states (if this were not the case, then we can preprocess the automaton in linear time and prune all unreachable states). If $\mathcal{A}$ is a deterministic finite state automaton (*DFA*), then we usually denote the unique initial state by q_0 and turn its transition relation E into a partial function δ from $Q \times \Sigma^*$ to Q defined by $\delta(q, \varepsilon) = q$ and $\delta(q, a \cdot u) = \delta(q', u)$ iff $(q, a, q') \in E$. It is worth noticing that, since the decision problems we are going to deal with are at least NLOGSPACE-hard and since any given automaton can be pruned using some simple NLOGSPACE reachability analysis, this assumption will have no impact on our complexity results.

Since automata can be viewed as directed (labeled) graphs, we inherit the standard definitions and constructions in graph theory. In particular, given an NFA $\mathcal{A} = (\Sigma, Q, E, I, F)$ and a state $q \in Q$, we denote by $\mathcal{C}(q)$ the *strongly connected component* (shortly, SCC) of $\mathcal{A}$ that contains all states mutually reachable from q. We say that a component C of $\mathcal{A}$ is *final* if it can reach a final state (possibly outside C). Given a set C of states of $\mathcal{A}$ (e.g., a SCC), we denote by $\mathcal{A}|C$ the automaton obtained by restricting $\mathcal{A}$ to the set C and by letting the new initial and final states be all and only the states in C (note that if C consists of a single transient state, then the language $\mathscr{L}(\mathcal{A}|C)$ recognized by the subautomaton $\mathcal{A}|C$ is empty). Finally, we denote by $\mathcal{D}ag(\mathcal{A})$ the directed acyclic (unlabeled) graph of the SCCs of $\mathcal{A}$ and by $\mathcal{D}ag^*(\mathcal{A})$ the graph obtained from the transitive closure of the edges of $\mathcal{D}ag(\mathcal{A})$.

Transducers. A (letter-to-word sequential) *transducer* is a device of the form $\mathcal{S} = (\Sigma, \Delta, Q, \delta, q_0, \Omega)$, where Σ is a finite input alphabet, Δ is a finite output alphabet, Q is a finite set of states, δ is a transition function from $Q \times \Sigma$ to $\Delta^* \times Q$, q_0 is an initial state, and Ω is a final output function from Q to Δ^*. For every input word $u = a_1 \dots a_n \in \Sigma^*$, there is one run of $\mathcal{S}$ of the form $q_0 \xrightarrow{a_1/v_1} q_1 \xrightarrow{a_2/v_2} \dots \xrightarrow{a_n/v_n} q_n \xrightarrow{\varepsilon/v_{n+1}}$, with $\delta(q_i, a_i) = (v_i, q_{i+1})$ for all $0 \leq i < n$ and $\Omega(q_n) = v_{n+1}$; in such a case, we define the *output* of $\mathcal{S}$ on u to be the word $\mathcal{S}(u) = v_1 v_2 \dots v_n v_{n+1}$. The only slightly non-standard feature here is the final output function, which allows an additional word to be added on at the end.

Transducers as above produce an output word immediately on reading an input character. We will also consider transducers with a bounded amount of "delay". A k-lookahead transducer, with $k \in \mathbb{N}$, is as above, but where the transition function δ now has input in $Q \times \Sigma \times (\Sigma_\perp)^k$, with $\Sigma_\perp = \Sigma \cup \{\perp\}$ and $\perp \notin \Sigma$. Given an input word u and a position $1 \leq i \leq |u|$ in it, we denote by $\vec{u}_i$ the $(k+1)$-character subword of $u \cdot \perp^k$ that starts at position i and ends at position $i + k$. The output of the k-lookahead transducer $\mathcal{S}$ on an input u of length n is the unique word $v = v_1 v_2 \dots v_n v_{n+1}$ for which there exists a sequence of states $q_0, ..., q_n$ satisfying $\delta(q_i, \vec{u}_i) = (v_i, q_{i+1})$, for all $1 \leq i \leq n$ and $\Omega(q_n) = v_{n+1}$. Clearly, a 0-lookahead transducer is simply a standard (letter-to-word sequential) transducer.

Logics. In this paper we look at languages defined by automata, and also consider *linear temporal logic LTL*, which utilizes the modal operators $\mathbf{X}$ (next) and $\mathbf{U}$ (until), along with boolean operators. Hereafter, we shall interpret LTL formulas on finite models only. This requires a careful use of the modal operators (for instance, the LTL formula $\mathbf{X}\mathtt{true}$ does not hold on singleton words). We also assume that the propositional variables of an LTL formula are precisely the symbols of the underlying alphabet (this means that two different propositional variables can not hold at the same position in a model).

Finally, recall that DFA and NFA are equally expressive DFA and NFA express exactly the regular languages while LTL expresses a strict subclass of them. NFA are exponentially more succinct than DFA.

III. Problem setting

Given two words $u \in \Sigma^*$ and $v \in \Delta^*$, we denote by $\mathcal{D}ist(u, v)$ the *Levenshtein distance* (henceforth, edit distance) between u and v, which is defined as the length of a shortest sequence s of edit operations (e.g., deleting a single character, modifying a single character, and inserting a single character) that transforms u into v [4].

Given two finite alphabets Σ and Δ, a *processor* is simply a function from Σ^* to Δ^*. Given a processor f, we refer to the edit distance between two words u and $f(u)$ as *the cost of f on the word u*. Given a language $R \subseteq \Sigma^*$ the *worst-case cost* of f over R is the supremum of the cost of f over all words in R (if the cost is unbounded, then we say that the worst-case cost is ω).

The general setting of a repair problem consists of two languages $R \subseteq \Sigma^*$ and $T \subseteq \Delta^*$, called the *restriction* and *target* languages, respectively. We would like to repair a string that is known to belong to the restriction language into a string in the target language. A processor f is a *repair strategy of R into T* if for every word $u \in R$, the output $f(u)$ is in T. We denote by $\mathcal{D}ist(R,T)$ the worst-case cost of an optimal repair strategy of R into T. It is easy to see that $\mathcal{D}ist(R,T) = \sup_{u \in R} \min_{v \in T} \mathcal{D}ist(u,v)$, since the best strategy is just to output on any $u \in R$ the word in T that is closest to u with respect to the edit distance.

The *bounded repair problem* is to decide, given (some representations of) languages R and T, whether $\mathcal{D}ist(R,T)$ is finite, that is, whether there is a repair strategy f of R into T and a natural number $n \in \mathbb{N}$ such that $\mathcal{D}ist(u, f(u)) \leq n$ for all $u \in R$. The *threshold problem* is to compute the exact value of $\mathcal{D}ist(R,T)$. Clearly, the languages R and T that form the input to the bounded repair problem must be finitely presented, for instance, in terms of machines or logical formulas. In this paper, we study the complexity of the bounded repair problem for input languages represented by means of the following formalisms: (i) deterministic finite state automata (DFA), (ii) non-deterministic finite state automata (NFA), and (iii) LTL formulas with only future modal operators (shortly, LTL formulas). We shall also consider special cases of the bounded repair problem, where the restriction language is assumed to be the universal language Σ^*, or where the repair strategy is implemented by a sequential transducer.

Streaming vs non-streaming. In its most general formulation, a repair strategy could be any function mapping words to edit words (in principle, such a function could even be not computable). However, we know from [3] that there is a dynamic programming algorithm that, given a word u and a target language T represented by a DFA, computes in linear time an optimal edit sequence s such that $s(u) \in T$. In particular, this shows that optimal repair strategies can be described by functions of fairly low complexity. Sometimes it is desirable to have repair strategies that are in even more limited classes. Perhaps the ideal case is when a strategy is realizable with a bounded memory one-pass algorithm, that is, using a (letter-to-word sequential) transducer. Recall that a letter-to-word transducer defines a word-to-word function (i.e., a processor); if this function is a repair strategy, we refer to the transducer as a *streaming repair strategy*. The idea is that any input word u from a restriction language should be repaired in an online way. Similarly, we can talk about a k-lookahead streaming repair strategy.

Accordingly, we define the bounded repair problem in the (k-lookahead) *streaming case* as the problem of deciding, given two languages R and T, whether there is a (k-lookahead) streaming strategy for repairing R into T with uniformly bounded cost. To stress the difference between the streaming and the non-streaming settings, we explicitly refer to the original problem as the bounded repair problem in the *non-streaming case*. The following example, due to Slawomir Staworko, illustrates the difference between the streaming and non-streaming setting:

Example 1. *Consider $R = (a+b)c^*(a^+ + b^+)$ and $T = ac^*a^+ + bc^*b^+$. One can get from R to T by only editing the initial letter: so the asymptotic cost is 0. A streaming strategy must commit as to changing the initial letter or leaving it be, and then will be forced to repair*

Costs in the streaming case. Note that, if we have a transducer $\mathcal{S}$ and a word $u = a_1 \dots a_n \in \Sigma^*$, then we can define the cost of $\mathcal{S}$ on u in two ways:

- letting $q_0 \xrightarrow{a_1/v_1} q_1 \xrightarrow{a_2/v_2} \dots \xrightarrow{a_n/v_n} q_n \xrightarrow{v_{n+1}}$ be the run of $\mathcal{S}$ on u, we define the *aggregate cost of $\mathcal{S}$ on u* to be the sum over all indices $1 \leq i \leq n$ of $\mathcal{D}ist(a_i, v_i)$, where $\mathcal{D}ist(a_i, v_i)$ is 1 if v_i is empty, $|v_i| - 1$ if a_i occurs in v_i, and $|v_i|$ otherwise;
- considering the transducer $\mathcal{S}$ as a processor, we define the (edit) *cost of $\mathcal{S}$ on u* to be simply the edit distance between u and the output $\mathcal{S}(u)$.

The first cost considers the distortions performed in producing the input from the output – it is equivalent to considering the transducer as producing edits rather than strings and counting the number of edits produced. The second cost is global and it considers only the output and not its production (clearly, this cost never exceeds the aggregate cost). These two models of cost can be very different in general. Consider a transducer $\mathcal{S}$ on the input alphabet $\Sigma = \{a, b\}$ that swaps a's and b's. On the string $u_n = (ab)^n$, the aggregate cost is $2n$ since $\mathcal{S}$ changes each letter, but the edit distance between u and $\mathcal{S}(u)$ (i.e., the edit cost of $\mathcal{S}$ on u in our sense) is only 2. Nevertheless, it will turn out that for the bounded repair problem it does not matter which model of cost we choose (see Theorem 3).

Special cases. We are also interested in a variant of the bounded repair problem where the restriction language is assumed to be a universal language of the form Σ^*. In this case, the input to the bounded repair problem consists of a restriction alphabet Σ and a target language T. We refer to this variant of the bounded repair problem as the *unrestricted case*.

A. Repair Problems, Automata, and Games

In the case of DFA, both the non-streaming and streaming problems correspond to special cases of prior problems studied in automata and games.

Non-streaming repair problems correspond to *distance automata*, while the streaming variant corresponds to *energy* *games*. We explain the correspondences in detail now. In both cases, we find that the results for the more general framework do not give tight bounds. .

Non-streaming repairs and distance automata. Intuitively, a *distance automaton* is a transducer $\mathcal{D}$ that receives as input a finite word w and outputs a corresponding cost $\mathcal{D}(w)$ in $\mathbb{N}\cup\{\infty\}$. Formally, the transitions of the distance automaton $\mathcal{D}$ are quadruples of the form (p,a,c,q), with p being a state in the restriction, a an input symbol, c a cost associated with the transition, and q a target state. The cost $\mathcal{D}(w)$ on input w is obtained by taking the minimum among the costs of the successful runs of $\mathcal{D}$ on w, where the cost of a successful run is defined as the sum of the costs of its transitions (we let $\mathcal{D}(w)=\infty$ if $\mathcal{D}$ admits no successful run on w).

The main problem that has been studied for distance automata is the *limitedness problem* which consists of deciding whether the cost function computed by a given distance automaton $\mathcal{D}$ is uniformly bounded on all accepted words. This problem was shown decidable by Hashigushi [5] and later in [6] was shown to be PSPACE-complete.

Distance automata have been related to edit-distance problems in several prior works – see the Related Work section for further discussion of the connections. Here we note only a simple reduction of the bounded repair problem to limitedness. Given two NFA $\mathcal{R}$ and $\mathcal{T}$, one can construct a distance automaton $\mathcal{D}$ that computes the cost of repairing any word from $\mathscr{L}(\mathcal{R})$ into a word from $\mathscr{L}(\mathcal{T})$. Let $\mathcal{R}=(\Sigma,Q,E,I,F)$ and $\mathcal{T}=(\Delta,Q',E',I',F')$ bet the two NFA for the restriction and the target languages. First of all, we associate with each symbol $a\in\Sigma$ a matrix $M(a)$ whose entries $M(a)[p,q]$ are indexed over the pairs of states p,q of $\mathcal{T}$ and give the minimum edit-distance between the symbol a and a word $v\in\Delta^*$ such that $p\xrightarrow{v}q$ (if q is not reachable from p, then we let $M(a)[p,q]=\infty$). We then define the distance automaton $\mathcal{D}$ as the quadruple $(\Sigma,Q\times Q',E\otimes M,I\times I',F\times F')$, where $E\otimes M$ is the set of all transitions of the form $\big((p,p'),a,c,(q,q')\big)$, with $a\in\Sigma$, $(p,a,q)\in E$, $p',q'\in Q'$, and $c=M(a)[p',q']$. It is easy to see that the cost function computed by $\mathcal{D}$ maps any word $u\in\mathscr{L}(\mathcal{R})$ (which is accepted by $\mathcal{D}$ too) to the cost of the best non-streaming repair of u into $\mathscr{L}(\mathcal{T})$. Moreover, the distance automaton $\mathcal{D}$ has size polynomial in the size of $\mathcal{R}$ and $\mathcal{T}$. Combining this reduction with the PSPACE upper bound for the limitedness problem, we see that the bounded repair problem for NFA is in PSPACE.

The same reduction technique can be applied to solve the bounded repair problem for DFA. In this case, however, the resulting complexity bound is not optimal: the bounded repair problem for DFA is in fact in co-NP (cf. Corollary 4).

Roughly speaking, the reason why the bounded repair problem for DFAs is easier than the limitedness problem for (unrestricted) distance automata is that the distance automata emerging from DFA repair problems are deterministic on the 0-cost moves. In addition to not giving tight bounds, approaches via distance automata give less insight into the problems. We invite the reader, for example, to compare the PSPACE upper bound that we derive from our characterization of repairability, Theorem 2, with the PSPACE upper bound given in [6].

Streaming repairs and energy games. Just as non-streaming repair problems can be seen within the framework of distance automata, bounded repair problems in the streaming setting are special cases of games on graph that pair qualitative (e.g., reachability) objectives and quantitative ones (e.g., maximization of payoff). An interesting family of such games is that of *energy games* studied by Chatterjee and Doyen [7], which are played on finite weighted arenas. The quantitative requirement is that the running sum of the weights (i.e., the energy) remains positive. The qualitative requirement could be either a safety condition, which makes sense for both finite and infinite plays or a parity, (e.g., Büchi) condition, which makes sense for infinite plays only Together, these requirements describe a game between an energy player, who wants to fulfill both quantitative and qualitative requirements, and her opponent. A variant of energy games allows the parameterization by an initial credit of energy; the higher the credit the more possibility for the energy player to win.

Theorem 2 in [7] shows that the problem of determining whether there is a finite initial credit so that the energy player has a winning strategy is in NP $\cap$ coNP (the exact complexity is however unknown). Theorem 3 in the same paper shows that same problem can be solved in time $\mathcal{O}\big(|E|\cdot d\cdot|Q|^{d+3}\cdot W\big)$, where $|E|$ denotes the number of edges of the arena, $|Q|$ denotes the number of nodes of the arena, W denotes the largest weight in absolute value of the arena, and d denotes the number of priorities of the parity condition. As a matter of fact, the latter complexity result implies that energy games with safety or Büchi conditions can be solved in polynomial time with respect to the size of the arena (provided that the weights are represented in unary).

One can easily reduce the bounded repair problem in the streaming setting, under the aggregate cost model for languages recognized by DFA, to the finite initial credit problem for energy safety games. Informally, the choice of the opponent in the energy game corresponds to the letters emitted by the restriction, while the edits correspond to choices of the energy player. Formally, we have a node in the arena for each pair of states of the restriction DFA $\mathcal{R}$ and of the target DFA $\mathcal{T}$ – call this node a “Restriction Player Node”. We also have a node for each combination of restriction state, target state, and letter played – call this a “Target Player Node”. The former represents the states reached by the restriction and target automata after parsing the unedited and edited words, while the latter adds the last letter emitted by the restriction. There is an edge of weight 0 going from a Restriction Player Node (p,p') to any Target Player Node (q,p',a), where (p,a,q) is a transition of the restriction DFA $\mathcal{R}$. Similarly, there is an edge of weight 0 going from a Target Player Node (q,p',a) to a Restriction Player Node (q,q'), where (p',a,q') is a transition of the target DFA $\mathcal{T}$. There are also edges of weight $-c$ from Target Player Nodes (q,p',a) to Restriction Player Nodes

(q,q'), provided that there is a word v at distance c from a (i.e., $\mathcal{D}ist(a,v)=c$) such that $\mathcal{T}$ can move from p' to q' consuming v. Finally, the safety condition requires the existence of a play that ends in a non-final state of $\mathcal{R}$ or in a final state of $\mathcal{T}$.

The above construction reduces the bounded repair problem in the non-streaming setting, under the aggregate cost model for DFA, to the problem of deciding whether a player can win an energy safety game with a finite initial credit. We observe that the size of the resulting arena is polynomial in the size of the restriction and target DFA and, moreover, the weights are bounded by the size of the target DFA. This gives a PTIME upper bound to the complexity of the bounded streaming repair problem for DFA.

Our characterization results (see Theorem 3) give analogous (tight) complexity bounds for languages recognized by DFA and moreover, prove that the bounded repair problem in the streaming setting is not sensitive to the models of aggregate/edit cost. They also provide tight bounds for special cases of the problem that can not naturally be captured in the setting of energy games. Our repair strategy can be seen as a special case of the notion of good-for-energy strategy, which is introduced in [7] to solve energy parity games. Intuitively, a good-for-energy strategy guarantees that the outcome of the play contains only cycles of non-negative energy (this excludes the possibility for the opponent to induce plays of arbitrary low energy). We characterize bounded repair using "good-for-repairer" component mappings, when a SCC C of the restriction automaton covers a SCC C' of the target automaton (see Section IV for a formal definition). In such a situation any word emitted by the restriction automaton with the component C does not need to be repaired in order to be recognized by the target automaton with the component C'.

Despite the connections mentioned above, many concepts and problems concerning repair do not have natural analogs in the game setting, and vice versa. For instance, in the game setting one could allow look-ahead for one player, but it is not as natural as in the repair setting. Moreover, while the aggregate cost metric fits the game setting naturally, our usual cost function does not. Conversely, the binary weights that are allowed in the game setting have no natural analog in the context of edits. Our characterization also allows us to easily isolated special cases of lower complexity that are not easily seen from the embedding into energy games.

IV. Characterizations of bounded repairability

The non-streaming case. We fix a restriction language R and a target language T and we assume that these languages are recognized by two NFA $\mathcal{R}$ and $\mathcal{T}$, respectively. Recall that $\mathcal{D}ag(\mathcal{R})$ is the directed acyclic graph of the SCCs of $\mathcal{R}$ and $\mathcal{D}ag^*(\mathcal{T})$ is the symmetric and transitive closure of $\mathcal{D}ag(\mathcal{T})$. Moreover, recall that we tacitly assume that all useless states are removed from both $\mathcal{R}$ and $\mathcal{T}$.

We say that a path $\pi = C_1 \dots C_n$ in $\mathcal{D}ag(\mathcal{R})$ is *covered* by a path $\pi' = C'_1 \dots C'_m$ in $\mathcal{D}ag^*(\mathcal{T})$ if we have (i) $n = |\pi| = |\pi'| = m$ and (ii) $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|C'_i)$ for all indices $1 \leq i \leq n$, namely, if the language recognized by the i-component along π is contained in the language recognized by the i-component along π'.

The following characterization reduces the bounded repair problem in the non-streaming case to the path matching problem in finite directed acyclic graphs.

Theorem 2. *Given two NFA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

1) *there is a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost,*
2) *every path in $\mathcal{D}ag(\mathcal{R})$ is covered by some path in $\mathcal{D}ag^*(\mathcal{T})$,*
3) *there is a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with worst-case cost at most $(1+|\mathcal{D}ag(\mathcal{R})|)\cdot|\mathcal{T}|$.*

The interesting directions are from 2) to 3) and from 1) to 2). For the first implication, if the coverability condition is satisfied, then we repair words w in $(\mathcal{R})$ by choosing any path π through $\mathcal{D}ag(\mathcal{R})$ taken by w, and looking at a covering path π'' in $\mathcal{D}ag(\mathcal{T})$. At the boundary points where w jumps from one component to the next in $\mathcal{R}$, we can insert small words that push the computation in $\mathcal{D}$ to the next component in π'; because these are strongly connected components, we can arrange a jump to any state in p'. Thus we can repair w with a small word.

The second implication is more complex, and is proven by contraposition. Assuming the negation of 2) we know that there is a path $\pi = C_1 \dots C_k$ of $\mathcal{D}ag(\mathcal{R})$ that is not covered. For each component $C_i \in \pi$ we construct a single word v_i that witnesses all non-containments of $\mathscr{L}(C_i)$ in components of $\mathcal{T}$. We then form for each n words w_n formed by concatenating n-fold iterations of the v_i: words $w_n = v'_0 v_1^n \dots v_k^n v'_k$ where the initial and final strings v'_0, v'_k are arranged to make sure the resulting word is in $\mathscr{L}(\mathcal{R})$. We argue that w_n requires at least n edits to be repaired into a word in $\mathscr{L}(\mathcal{T})$.

The streaming case. We now modify Theorem 2 to give a characterization of the streaming repair problem, adding in a game setting.

Fix two DFAs $\mathcal{R}$ and $\mathcal{T}$ recognizing the restriction and target languages. We associate with the DFA a *reachability game* between two players, Adam and Eve, on a suitable arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, defined in terms of the SCCs of $\mathcal{R}$ and $\mathcal{T}$. The idea underlying this game is as follows: during Adam's construction of a path π in $\mathcal{D}ag(\mathcal{R})$, Eve has to provide a construction of a corresponding path $f(\pi)$ in $\mathcal{D}ag^*(\mathcal{T})$ that covers π; moreover, the resulting function f must satisfy the following condition: if $\pi \cdot C$ is an extension of the path π in $\mathcal{D}ag(\mathcal{R})$ by a single SCC, then either $f(\pi \cdot C)$ coincides with $f(\pi)$ or it is an extension of $f(\pi)$ by a single SCC, namely, $f(\pi \cdot C)$ is of the form $f(\pi)\cdot D$.

Formally, the nodes of the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ where player Adam (resp., Eve) nodes are the pairs of the form (C,D) (resp., (D,C)), where C is a SCC of $\mathcal{R}$ and D is a SCC of $\mathcal{T}$. The edges of the arena connect Adam's nodes (C,D) to Eve's nodes (D,C') where (C,C') is an edge of $\mathcal{D}ag(\mathcal{R})$

and, similarly, Eve's nodes (D, C) to Adam's nodes (C, D') where (D, D') is an edge of $\mathcal{D}ag^*(\mathcal{T})$ and, in addition, $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D')$. The initial node is an Eve node (D_0, C_0), where C_0 is the SCC of the initial state of $\mathcal{R}$ and D_0 is the SCC of the initial state of $\mathcal{T}$. The last player who moves wins. Intuitively, Adam's objective is to reach a node (C, D) where Eve cannot respond with any move, conversely, Eve's objective is to reach a node (D, C) where Adam cannot respond with any move. As usual, we say that a player has a winning strategy on the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ if he/she can win the reachability game on $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ independently of choices of the other player.

The following characterization reduces the bounded repair problem in the streaming case to the problem of determining the winner of a suitable reachability game. It also shows that, quite surprisingly, the bounded repair problem in the streaming setting is not sensitive to the notions of transducer with/without lookahead and to the models of aggregate/edit cost.

Theorem 3. *Given two DFA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

1) *there is a k-lookahead streaming strategy, for some $k \in \mathbb{N}$, that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded edit cost,*
2) *Eve has a winning strategy for the reachability game on $\mathcal{A}_{\mathcal{R},\mathcal{T}}$,*
3) *there is a 0-lookahead streaming strategy that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with worst-case* aggregate *cost at most $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$.*

V. Complexity results in the non-streaming case

In this section, we study the complexity of the bounded repair problem and the threshold problem in the non-streaming setting.

A. The bounded repair problem

We begin by analyzing the complexity in the case of languages recognized by non-deterministic finite automata.

NFA. Theorem 2 gives a straightforward PSPACE algorithm that solves the bounded repair problem between two NFA $\mathcal{R}$ and $\mathcal{T}$ in this setting: the algorithm first guesses universally a path $\pi = C_1...C_n$ in $\mathcal{D}ag(\mathcal{R})$, then it guesses existentially a path $\pi' = C'_1...C'_n$ of the same length in $\mathcal{D}ag^*(\mathcal{T})$, and finally it checks the containment of the subautomaton $\mathcal{R}|C_i$ in the subautomaton $\mathcal{T}|C'_i$ for all indices $1 \leq i \leq n$. Together with the PSPACE lower bound for the problem proven later (see Corollary 20), we obtain:

Corollary 4. *The bounded repair problem in the non-streaming case, where the restriction and target languages are represented by NFA, is PSPACE-complete.*

DFA. The same characterization result can be used to solve the problem when the restriction language is represented by an NFA and the target language is represented by a DFA. In this case, we can take advantage of the determinism to show that the problem turns out to be coNP-complete. Intuitively, the complexity upper bound follows from the observation that containment of languages recognized by SCCs of DFA is decidable in PTIME (even if the successful runs can start from arbitrary states inside the SCCs) and that the above mentioned coverability problem for paths of SCCs is in coNP (indeed, the coverability problem can be solved using a variant of the online subset construction over the graph of the SCCs of the target automaton). The complexity lower bound follows from a reduction from the validity problem for propositional formulas in disjunctive normal form (i.e., the dual of the SAT problem): the idea is to encode in the restriction language all the possible valuations for the propositional variables (here some redundancy is needed to forbid the repair strategy from modifying these valuations) and then restrict the target language to consist only of encodings of valuations that satisfy at least one clause of the formula.

Theorem 5. *The bounded repair problem in the non-streaming case, where the restriction language is represented by an NFA and the target language is represented by a DFA, is in coNP and it is coNP-hard already for languages represented by DFA.*

Before turning to the complexity of the bounded repair problem for languages specified by LTL formulas, we briefly outline some parameterized complexity results in the automaton case. Quite surprisingly, if we fix either the restriction language or the target language, then the bounded repair problem in the non-streaming case becomes tractable. We first consider the case where the restriction automaton is fixed and the target automaton is a DFA provided as input to the problem. Using arguments similar to the previous coNP upper bound, one can show that the bounded repair problem between a fixed restriction language and the target language recognized by a given DFA is in PTIME:

Proposition 6. *Let R be a fixed restriction language. The problem of deciding, given a DFA $\mathcal{T}$, whether there is a non-streaming repair strategy of R into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost is in PTIME.*

Similarly, the bounded repair problem is tractable when we fix the target automaton and let the restriction automaton be an NFA provided as input to the problem:

Proposition 7. *Let T be a fixed target language. The problem of deciding, given an NFA $\mathcal{R}$, whether there is a non-streaming repair strategy of $\mathscr{L}(R)$ into T with uniformly bounded cost is in PTIME.*

LTL. We conclude the section by analyzing the complexity of the bounded repair problem where languages defined by LTL formulas are involved. We first consider the problem where both the restriction language R and the target language T are defined by some LTL formulas ϕ and ψ. It is not difficult to see that this problem is in coNEXPTIME. Indeed, one can use standard automaton-based techniques to construct, in exponential time, two *DFA* $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$ that recognize the *reversals* $\overleftarrow{R}$ and $\overleftarrow{T}$ of the languages R and T. Since, in the non-streaming setting, the cost of repairing R into T is the same as the cost of repairing $\overleftarrow{R}$ into $\overleftarrow{T}$, one can exploit Theorem 5 to solve the bounded repair problem on the DFA $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$. As for the complexity lower bound, one can reduce the problem of deciding the non-existence of a tiling of an exponential square grid – this problem is known to be coNEXPTIME-complete [8] – to the problem of deciding the existence of a repair strategy of uniformly bounded cost between two regular languages defined by suitable LTL formulas. The general idea of such a reduction is to let the formula for the restriction language encode all candidate tilings and the formula for the target language check that none of them is correct.

Theorem 8. *The bounded repair problem in the non-streaming case, where the restriction and target languages are represented by LTL formulas, is coNEXPTIME-complete.*

The bounded repair problem becomes easier when it involves repairs of languages recognized by NFA into language defined by LTL formulas, or, symmetrically, repairs of languages defined by LTL-formulas into languages recognized by NFA:

Theorem 9. *The bounded repair problem in the non-streaming case, where the restriction language R is represented by an NFA and the target language T is represented by an LTL formula, is in PSPACE.*

Theorem 10. *The bounded repair problem in the non-streaming case, where the restriction language R is represented by an LTL formula and the target language T is represented by an NFA, is in PSPACE.*

B. The threshold problem

We now consider the problem of calculating the exact cost. In the case of DFA, we know from Theorem 5 that we can determine whether the repair cost is finite or infinite in coNP. Furthermore, Theorem 2 tells us that if the cost is finite it must be bounded by a polynomial in the input size.

Thus, to determine the exact repair cost in the case where it is finite, it suffices to test whether the cost is above or below a given threshold k in unary, since then we can try every k below the polynomial bound. Perhaps surprising, this problem is harder than the finiteness problem, although still within polynomial space:

Theorem 11. *The problem of determining, given k and two languages R and T recognized by DFA, whether $\mathcal{D}ist(R,T)$ is above k, is PSPACE-complete. The same holds when R and T are given as an NFA.*

In the case of LTL, it is not a priori even clear how to compute the distance of a single word to a formula. However, this can be shown to be in PSPACE (see the appendix). In the general case of two LTL formula we get an exponential blow-up over the automata case, as expected:

Theorem 12. *The problem of determining, given k and two languages R and T defined by LTL formulas, whether $\mathcal{D}ist(R,T)$ is above k, is EXPSPACE-complete.*

VI. Complexity results in the streaming case

A. The bounded repair problem

DFA. Let us consider two DFA $\mathcal{R}$ and $\mathcal{T}$. The characterization of Theorem 3 shows that the problem of deciding the existence of a streaming repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost amounts at solving a reachability game over a suitable (acyclic) arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. In particular, we observe that the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ can be computed from $\mathcal{R}$ and $\mathcal{T}$ in polynomial time (recall that checking containment of languages recognized by SCCs of automata is in PTIME). Moreover, it is known that the problem of deciding the winner of reachability games over acyclic graphs is PTIME-complete [9]. This shows that the bounded repair problem for DFA in the streaming case is PTIME-complete:

Corollary 13. *The bounded repair problem in the streaming case, where the restriction and target languages are represented by DFA, is PTIME-complete.*

It is worth remarking that the complexity of the bounded repair problem for DFA in the streaming setting is lower than the analogous problem in the non-streaming setting (indeed Theorem 5 shows that the latter problem is coNP-complete). This will be in contrast with the complexity results for languages defined by LTL formulas, where the non-streaming setting becomes more difficult than the streaming setting (compare Theorem 5 and Theorem 16).

NFA. When both restriction and target are NFA we are not able to provide tight complexity bounds, thus we only claim that the complexity of the bounded repair problem for NFA is between PSPACE and EXPTIME (the lower bound follows from Corollary 20 and the upper bound from the standard subset construction on NFA):

Corollary 14. *The bounded repair problem in the streaming case, where the restriction and target languages are represented by NFA, is in EXPTIME and it is PSPACE-hard.*

In the case where either the restriction or target language is a DFA, we have tight bounds:

Theorem 15. *The bounded repair problem in the streaming case, where the restriction language is a DFA and the target language is an NFA is PSPACE-complete. The same result holds when the target language is DFA and the restriction is an NFA.*

In the case where the target is a DFA but the restriction is an NFA, one could also prove an EXPTIME bound on the bounded repair problem via reduction to *energy games with imperfect information* (studied by Degorre et. al. in [10]). However, this does not give a tight bound, and we do not know how to extend the reduction to non-deterministic targets.

LTL restriction and target. We now turn to the complexity of the bounded repair problem in the streaming case, where both restriction and target languages are represented by LTL formulas. By following standard constructions in automata theory, one can translate any pair of LTL formulas ϕ and ψ

into DFA $\mathcal{R}$ and $\mathcal{T}$ that have size doubly exponential in the size of the formulas ϕ and ψ and that recognize the same languages defined by ϕ and ψ. This gives a straightforward 2EXPTIME upper bound to the complexity of the bounded repair problem. As for the complexity lower bound, we can reduce the problem of deciding the winner of tiling game over an exponential square grid – this problem is known to be EXPSPACE-complete [8] – to the problem of deciding the existence of a streaming repair strategy of uniformly bounded cost between the languages defined by suitable LTL formulas (the idea of such a reduction is similar to the coNEXPTIME-hardness proof of Theorem 8):

Theorem 16. *The bounded repair problem in the streaming case, where the restriction and target languages are represented by LTL formulas, is in 2EXPTIME and is EXPSPACE-hard.*

B. The threshold problem and constructing streaming repairs

For the streaming case, if we consider streaming repair strategies with aggregated cost, the threshold problem maintains its PTIME complexity. Further, one can easily reduce this threshold problem to a reachability game over a suitable arena.

Theorem 17. *The problem of determining, given k and two languages R and T recognized by DFA, whether one can repair R into T in the streaming case with aggregate cost at most k, is in PTIME.*

In fact, it follows from the reduction that *one can efficiently compute the optimal streaming repair* that satisfies a given threshold from the previous reduction. This is because we can construct a streaming repair strategy that satisfies a given threshold by finding a winning strategy for Eve in the reachability game. Finding such a strategy is well-known to be in PTIME.

Corollary 18. *Let R and T be the restriction and target languages specified by DFA. If R is streaming repairable into T with aggregate cost at most k, then an optimal streaming repair strategy of R into T with aggregate cost at most k can be computed in PTIME.*

Note that in the above we deal with the aggregate cost; the example from Section III shows that this cost can differ from the edit cost, while our characterization theorem shows that one is finite iff the other is. We do not know if finding the exact edit cost is tractable.

VII. Special cases: unrestricted repair problems

We now consider a special case of the bounded repair problem, namely, the variant where the restriction language is assumed to be a universal language Σ^* and the target language T is represented by means of finite state automata.

The following result adapts the characterization theorems given in Section IV to give a necessary and sufficient condition for the existence of a repair strategy with uniformly bounded cost from a universal language Σ^* to the language recognized by an NFA $\mathcal{T}$. This result also shows that there is no difference between the non-streaming and the streaming settings when the restriction language is universal (indeed it can be viewed as a special case of both Theorem 2 and Theorem 3).

Corollary 19. *Given an alphabet Σ and an NFA $\mathcal{T}$, the following conditions are equivalent*

1) *there is a repair strategy of Σ^* into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost,*
2) *$\mathcal{T}$ has a SCC C such that $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|C)$,*
3) *there is a 0-lookahead streaming strategy that repairs Σ^* into $\mathscr{L}(\mathcal{T})$ with worst-case aggregate cost at most $2|\mathcal{T}|$.*

Using the above characterization, one can easily devise an NLOGSPACE algorithm that solves the bounded repair problem for DFA in the unrestricted (streaming or non-streaming) case. Indeed, if the target automaton $\mathcal{T}$ is a DFA and C is a component of $\mathcal{T}$, then we have $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|C)$ iff for every symbol $a \in \Sigma$ and every state q in C, $\mathcal{T}$ contains a transition of the form (q, a, q'), with $q' \in C$. Checking this property amounts to performing a standard NLOGSPACE reachability analysis over $\mathcal{T}$. Conversely, NLOGSPACE-hardness follows from the fact that the emptiness problem for DFA is reducible to the bounded repair problem: given a DFA $\mathcal{A}$ over an alphabet Σ, we have that $\mathscr{L}(\mathcal{A}) \neq \emptyset$ iff Σ^* is repairable into $\mathscr{L}(\mathcal{A}')$ with uniformly bounded cost, where $\mathcal{A}'$ is a DFA that recognizes the language $\Sigma^* \cdot \mathscr{L}(\mathcal{A})$ and that can be computed from $\mathcal{A}$ using simple logarithmic-space constructions.

In a similar way, one can show that the bounded repair problem for NFA in the unrestricted case is PSPACE-complete. This complexity result follows from Corollary 19 and from suitable reductions to/from the universality problem for NFA. Indeed, checking whether a target NFA $\mathcal{T}$ has a final SCC C such that $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|C)$ is equivalent to the problem of deciding whether Σ^* is repairable into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost, and it is clearly reducible to the universality problem for NFA. As for the PSPACE-hardness, we observe that a given NFA $\mathcal{A}$ recognizes the universal language Σ^* iff $\Sigma \uplus \{\#\})^*$ is repairable into $\mathscr{L}(\mathcal{A}_\#)$ with uniformly bounded cost, where $\#$ is a fresh separator symbol and $\mathcal{A}_\#$ is the automaton that recognizes the language $\{u_1 \# u_2 \# \ldots \# u_n : n \in \mathbb{N}, u_1, u_2, \ldots, u_n \in \mathscr{L}(\mathcal{A})\}$ (note that $\mathcal{A}_\#$ can be computed from $\mathcal{A}$ in linear time).

We thus conclude the following:

Corollary 20. *The bounded repair problem in the unrestricted case, where the target languages are represented by DFA (resp., NFA) is NLOGSPACE-complete (resp., PSPACE-complete).*

Another consequence of Corollary 19 is the following. Suppose that a target language T is recognized by a DFA $\mathcal{T}$ that is *complete* over the target alphabet Δ, namely, for every symbol $a \in \Delta$ and every state p of $\mathcal{T}$, $\mathcal{T}$ contains a transition from p labeled by a. Let us consider a restriction alphabet Σ contained in Δ and suppose that Σ^* is *not* repairable into T with uniformly bounded cost. Let us consider a SCC C

of $\mathcal{T}$ that is reachable from the initial state and terminal, namely, with no outgoing edges. We know that C does not contain any final state (otherwise, C would be a final SCC and hence, by Corollary 19, Σ^* would be repairable into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost). In this case, however, the same component C in the complement DFA $\mathcal{T}^{\complement}$ would be both final and terminal, and hence Σ^* would be repairable into $\mathscr{L}(\mathcal{T}^{\complement})$ $(= \Delta^* \setminus T)$ with uniformly bounded cost. This shows that for every restriction alphabet $\Sigma \subseteq \Delta$ and every regular language $T \subseteq \Delta^*$, there exists a (streaming) repair strategy from Σ^* to T or from Σ^* to the complement $\Delta^* \setminus T$ (sometimes, both cases happen).

Corollary 21. *Given an alphabet Σ and a regular language $T \subseteq \Delta^*$, with $\Sigma \subseteq \Delta$, one of the following two cases (possibly both) holds:*

1) Σ^* is repairable into T with uniformly bounded cost,

2) Σ^* is repairable into $\Delta^* \setminus T$ with uniformly bounded cost.

We now turn to the complexity of the bounded repair problem in the unrestricted case, but where the target languages are represented by LTL formulas. We claim that problem is PSPACE-hard for LTL formulas. This complexity lower bounds follows from arguments similar to the automaton-based setting, namely, from a reduction of the satisfiability problem for LTL formulas, which is known to be PSPACE-hard [11]. As for the complexity upper bound, we claim that the problem for LTL formulas is in PSPACE (thus PSPACE-complete). Indeed, given an LTL formula ψ defining a target language T, one can compute in polynomial time a symbolic representation of a DFA $\mathcal{T}'$ that recognizes the *reversal* T' of T. Moreover, one can perform standard reachability analysis on the symbolic representation of $\mathcal{T}$ in polynomial space. Finally, we observe that Σ^* is repairable into T with uniformly bounded cost iff Σ^* is repairable into T' with uniformly bounded cost. This shows that the bounded repair problem in the unrestricted case for LTL formulas is in PSPACE.

Corollary 22. *The bounded repair problem in the unrestricted case, where the target languages are represented by LTL formulas, is PSPACE-complete.*

VIII. Towards infinite words

We briefly discuss a natural generalization of our characterization result for the bounded repair problem in the streaming setting.

Recall that Theorem 2 reduces the bounded non-streaming repair problem to the problem of deciding the property of coverability between paths of SCCs in the DAGs of the restriction and target automata. If we turn to languages of infinite words recognized by non-deterministic Büchi automata (NBA), then the characterization result is similar. There is however a slight complication due to the acceptance condition in the infinite case.

First of all, we modify the notation for the sub-automata obtained by restricting to SCCs. As in the previous cases for NFA, given an NBA $\mathcal{B}$ and a SCC C of it, we write $\mathcal{R}|C$ to denote the usual NFA obtained by restricting $\mathcal{B}$ to the states in C and by letting them be both initial and final states. We also write $\mathcal{R}|^{\omega}C$ to denote the NBA obtained by restricting $\mathcal{B}$ to the set of states in C and by letting them be initial (we do keep instead the final states as in $\mathcal{B}$).

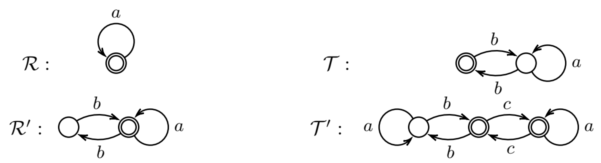

Figure 1: Some non-deterministic Büchi automata.

To understand why we introduce the two variants $\mathcal{B}|C$ and $\mathcal{B}|^{\omega}C$ of sub-automata, it is worth looking at the following examples. Let $\mathcal{R}$ and $\mathcal{T}$ be the single-component NBA depited at the top of Figure 1 (all states are initial, the final states are those with double circles). Observe that, when we view $\mathcal{R}$ and $\mathcal{T}$ as NFA, we have $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|C')$, and hence, by Theorem 2, $\mathcal{D}ist\big(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})\big) < \omega$. However, when we view $\mathcal{R}$ and $\mathcal{T}$ as NBA, we have $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|C')$, but $\mathcal{D}ist\big(\mathscr{L}^{\omega}(\mathcal{R}), \mathscr{L}^{\omega}(\mathcal{T})\big) = \omega$. On the other hand, if we consider the NBA $\mathcal{R}'$ and $\mathcal{T}'$ at the bottom of Figure 1, and we denote by C and C' be their unique SCCs, then we have that $\mathscr{L}^{\omega}(\mathcal{R}'|^{\omega}C) \nsubseteq \mathscr{L}^{\omega}(\mathcal{T}'|^{\omega}C')$, but $\mathcal{D}ist\big(\mathscr{L}^{\omega}(\mathcal{R}'), \mathscr{L}^{\omega}(\mathcal{T}')\big) < \omega$. The above examples suggest that we should use both variants of sub-automata for establishing a characterization result for bounded non-streaming repairability of languages recognized by NBA.

We now turn to the generalization of the notion of coverability. Given two NBA $\mathcal{R}$ and $\mathcal{T}$, a path π of length k in $\mathcal{D}ag(\mathcal{R})$, and a *set of paths* Π' in $\mathcal{D}ag^*(\mathcal{T})$, we say that π is *Büchi-covered* by Π' iff

i) all paths in Π' have length precisely $k+1$,

ii) $\mathscr{L}(\mathcal{R}|\pi(i)) \subseteq \bigcap_{\pi' \in \Pi'} \mathscr{L}(\mathcal{T}|\pi'(i))$ for all indices $i < k$,

iii) $\mathscr{L}^{\omega}(\mathcal{R}|^{\omega}\pi(k)) \subseteq \bigcap_{\pi' \in \Pi'} \mathscr{L}(\mathcal{T}|\pi'(k)) \cdot \bigcup_{\pi' \in \Pi'} \mathscr{L}^{\omega}(\mathcal{T}|^{\omega}\pi'(k+1))$.

The characterization theorem for bounded non-streaming repairability of NBA-recognizable languages is as follows:

Theorem 23. *Given two NBA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

1) there is a repair strategy of $\mathscr{L}^{\omega}(\mathcal{R})$ into $\mathscr{L}^{\omega}(\mathcal{T})$ with uniformly bounded cost,

2) every path in $\mathcal{D}ag(\mathcal{R})$ is Büchi-covered by a set of paths in $\mathcal{D}ag^*(\mathcal{T})$,

3) there is a repair strategy of $\mathscr{L}^{\omega}(\mathcal{R})$ into $\mathscr{L}^{\omega}(\mathcal{T})$ with worst-case cost at most $(2 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$.

We omit the proof of this theorem, which is almost identical to that of Theorem 2, and we instead invite the reader to check that the characterization for the infinite-word case is consistent with the examples that we gave above. As a matter of fact, the above characterization result easily yields a PSPACE upper bound for the bounded non-streaming repair problem between languages recognized by NBA.

IX. Related Work and Conclusions

In this work we have investigated language repair in the most basic setting of words. Our results are summarized in the tables below – in the non-streaming setting our bounds are tight (indicated by a single class), while in the streaming setting we have several gaps (where a cell gives lower and upper bounds).

	fixed	DFA	NFA	LTL
fixed	Const	PTIME	PSPACE	PSPACE
DFA	PTIME	CoNP	PSPACE	PSPACE
NFA	PTIME	PTIME	PSPACE	PSPACE
LTL	PSPACE	PSPACE	PSPACE	CoNEXP

Table I: Complexity of bounded non-streaming repair

	fixed	DFA	NFA	LTL
fixed	Const	PTIME	PSPACE	PSP, EXPSP
DFA	PTIME	PTIME	PSPACE	PSP, EXPSP
NFA	PSPACE	PSPACE	PSP, EXP	PSP, 2EXP
LTL	PSP, EXPSP	PSP, EXPSP	PSP, 2EXP	EXPSP, 2EXP

Table II: Complexity of bounded streaming repair

We omit the corresponding table for computing the exact cost: in the case of non-streaming repair we can derive tight bounds in all cases, and also in the case of streaming repair for aggregate cost. In the latter case we can also know the complexity of computing the optimal stream repair processor.

This paper deals with finding a uniform bound on the number of repairs needed over all instances. However, in a parallel work we have isolated the complexity of determining the repair strategy that is optimal for *normalized cost* of repair, where normalized cost divides number of repairs by the string length (in particular, it is always bounded).

We will be looking at extensions to trees next: the initial motivation for this work was to use repair policies as a way to correct input XML documents known to satisfy a source schema to result documents satisfying a target. Obviously, the notion of edit must be extended to the tree context, and the class of transducers considered must likewise be modified.

We have focused on the case of finite words in this paper, but infinite words raise many new issues. In the case of infinite words in a streaming setting, one can look for strategies that allow finitely-many edits per word, without a uniform bound, and likewise look for strategies with "continuous" (but not uniformly-bounded) look-ahead. This last issue has been investigated for purely qualitative games by Holtmann et. al. [12].

Related work on edit distance of languages. The problem of finding the minimal distance of a string to a regular language was first considered by Wagner in [3], who showed that the problem could be solved by adapting the dynamic programming approach to edit distance, giving a polynomial time algorithm.

Several authors have extended the definition to deal with distances between languages. Mohri [13] defines a distance function between two sets of strings, and more generally between string distributions: in the case of languages, this is the minimum distance between two strings in the two respective languages, which is appropriate for many applications. Konstantinidis [14] focuses on the minimum distance between distinct strings within the same language, giving tractable algorithms for computing it. Our notion of "cost" is quite distinct from this, since it is asymmetric in the two languages, focusing on the maximum of the distance of a string in one language to the other language.

Grahne and Thomo [15] consider a related problem of "approximate containment" of regular expressions. Expressions are evaluated with respect to an edge-labeled graph and are given a numerical semantics by a "distortion" – a generalization of the notion of edit distance. Approximate containment of T_1 and T_2 means, roughly speaking, that for every input graph R and every word w generated by R, the distance to target T_1 is bounded by the distance to T_2. Grahne and Thomo also study "k-containment" (distance to T_1 is at most k more than T_2) and "approximate-containment" (k-containment for some k), relying primarily on a reduction to the limitedness problem for distance automata. Their problem differs in several fundamental respects from ours: they are interested in bounding the difference over all words, not just the worst case; in addition they quantify over all restrictions (databases, in their terminology).

An entire line of research in XML data management has dealt with comparisons and matching algorithms between schema languages; many of these lift edit distance between trees to the level of schemas (i.e.g languages) – see, for example, [16]. However the lifting is done by looking at the syntactic structure of the schema description, rather than at the instance level (distance between documents/trees in each schema).

References

[1] M. Arenas, L. Bertossi, and J. Chomicki, "Consistent query answers in inconsistent databases," in *PODS*, 1999, pp. 68–79.

[2] F. Afrati and P. Kolaitis, "Repair checking in inconsistent databases: Algorithms and complexity," in *ICDT*, 2009, pp. 31–41.

[3] R. Wagner, "Order-n correction for regular languages," *CACM*, vol. 17, no. 5, pp. 265–268, 1974.

[4] R. Wagner and M. Fischer, "The string-to-string correction problem," *JACM*, vol. 21, no. 1, pp. 168–173, 1974.

[5] K. Hashiguchi, "Improved limitedness theorems on finite automata with distance functions," *Theor. Comp. Sci.*, vol. 72, no. 1, pp. 27–38, 1990.

[6] H. Leung and V. Podolskiy, "The limitedness problem on distance automata: Hashiguchi's method revisited," *Theor. Comp. Sci.*, vol. 310, pp. 147–158, 2004.

[7] K. Chatterjee and L. Doyen, "Energy parity games," in *ICALP*, 2010, pp. 599–610.

[8] P. Van Emde Boas, "The convenience of tilings," in *Complexity, Logic and Recursion Theory*, vol. 187, 1997, pp. 331–363.

[9] C. Papadimitriou, *Computational Complexity*. Addison-Wesley Longman Publishing Co., Inc., 1994.

[10] B. Puchala and R. Rabinovich, "Parity games with partial information played on graphs of bounded complexity," in *MFCS*, 2010, pp. 604–615.

[11] A. Sistla and E. Clarke, "The complexity of propositional linear temporal logics," *JACM*, vol. 32, no. 3, pp. 733–749, 1985.

[12] M. Holtmann, L. Kaiser, and W. Thomas, "Degrees of lookahead in regular infinite games," in *FOSSACS*, 2010, pp. 252–266.

[13] M. Mohri, "Edit-distance of weighted automata: general definitions and algorithms," *Int'l Journal of Foundations of Comp. Sci.*, vol. 14, no. 6, pp. 957–982, 2003.

[14] S. Konstantinidis, “Computing the edit distance of a regular language,” *Inf. and Comp.*, vol. 205, no. 9, pp. 1307–1316, 2007.
[15] G. Grahne and A. Thomo, “Query answering and containment for regular path queries under distortions,” in *FOIKS*, 2004, pp. 98–115.
[16] H. Do and E. Rahm, “COMA - a aystem for flexible combination of schema matching approaches,” in *VLDB*, 2002, pp. 610–621.

APPENDIX

A. *Proofs for Section IV (Characterizations of bounded repairability)*

Theorem 2. *Given two NFA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

1) *there is a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost,*
2) *every path in $\mathcal{D}ag(\mathcal{R})$ is covered by some path in $\mathcal{D}ag^*(\mathcal{T})$,*
3) *there is a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with worst-case cost at most $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$.*

Proof: Let $\mathcal{R} = (\Sigma, Q, E, I, F)$ and $\mathcal{T} = (\Delta, Q', E', I', F')$. We first prove the implication from *2)* to *3)*; later we will prove the implication from *1)* to *2)* (the implication from *3)* to *1)* is trivial and thus omitted).

Suppose that every path in $\mathcal{D}ag(\mathcal{R})$ is covered by some path in $\mathcal{D}ag^*(\mathcal{T})$. We have to prove the existence of a repair strategy of $\mathcal{R}$ into $\mathcal{T}$ with worst-case cost at most $(1+|\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$. Let us fix a generic word u in the restriction language $\mathscr{L}(\mathcal{R})$ and let ρ be a successful run of $\mathcal{R}$ on u. Clearly, the run ρ of $\mathcal{R}$ induces a path π in $\mathcal{D}ag(\mathcal{R})$, which is defined as the sequence of SCCs of $\mathcal{R}$ visited by ρ. Suppose that $\pi = C_1 C_2 \ldots C_n$. Accordingly, we factorize the word u into the sequence of subwords $u_1, a_1, u_2, a_2, \ldots, u_n$, where $u_i \in \mathscr{L}(\mathcal{R}|C_i)$ for all $1 \leq i \leq n$ and $a_i \in \Sigma$ for all $1 \leq i \leq n-1$. From the assumption on the coverability of the paths in $\mathcal{D}ag(\mathcal{R})$, we know that there is a path $\pi' = C'_1 C'_2 \ldots C'_n$ in $\mathcal{D}ag^*(\mathcal{T})$ that covers π, namely, such that $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|C'_i)$ for all $1 \leq i \leq n$. This shows that each subword u_i of u, with $1 \leq i \leq n$, belongs to the language $\mathscr{L}(\mathcal{T}|C'_i)$.
We can now construct inductively a corresponding word $f(u) = v_0 \cdot u_1 \cdot v_1 \cdot u_2 \cdot \ldots \cdot u_n \cdot v_n$ that is accepted by $\mathcal{T}$. We recall $\pi' = C'_1 \ldots C'_n$ is a path in $\mathcal{D}ag^*(\mathcal{T})$ and that the automaton $\mathcal{T}$ is pruned (namely, it does not contain useless states). This means that all states in C'_1 can be reached from some initial state of $\mathcal{T}$ and, similarly, all states in C'_n can reach some final state. In particular, since $u_1 \in \mathscr{L}(\mathcal{T}|C'_1)$, we know that there exist an initial state q_0 of $\mathcal{T}$, two states p_1 and q_2 in C'_1, and a word $v_0 \in \Delta^*$ such that $\mathcal{T}$ admits a run of the form $q_0 \xrightarrow{v_0} p_1 \xrightarrow{u_1} q_1$. Moreover, without loss of generality, we can assume that the length of v_0 is bounded by the number of states of $\mathcal{T}$. As for the inductive step, we assume that the words $v_0, \ldots, v_{i-1}$, with $0 \leq i \leq n-1$, are defined and that $\mathcal{T}$ admits a run on $v_0 \cdot u_1 \cdot \ldots \cdot v_{i-1}$ from the initial state q_0 to a state $q_i \in C'_i$. Since every state in C'_{i+1} is reachable from every state in C'_i, we know that there is a word v_i, with $|v_i| \leq |\mathcal{T}|$, and two states p_{i+1} and q_{i+1} in C'_{i+1} such that $\mathcal{T}$ admits a run of the form $q_i \xrightarrow{v_i} p_{i+1} \xrightarrow{u_{i+1}} q_{i+1}$. As for the final step, we assume that $v_0, \ldots, v_{n-1}$ are defined and that $\mathcal{T}$ admits a run on $v_0 \cdot u_1 \cdot \ldots v_{n-1} \cdot u_n$ from the initial state q_0 to a state $q_n \in C'_n$. Using arguments similar to the previous ones and the fact that C'_n is a final SCC, we derive the existence of a word v_n, with $|v_n| \leq |\mathcal{T}|$, and a final state p_{n+1} of $\mathcal{T}$ such that $\mathcal{T}$ admits a run of the form $q_n \xrightarrow{v_n} p_{n+1}$. Putting this all together, we obtain the existence of a successful run of $\mathcal{T}$ of the form

$$q_0 \xrightarrow{v_0} p_1 \xrightarrow{u_1} q_1 \xrightarrow{v_1} p_2 \xrightarrow{u_2} \ldots p_n \xrightarrow{u_n} q_n \xrightarrow{v_n} p_{n+1}$$

and hence the word $f(u) = v_0 \cdot u_1 \cdot v_1 \cdot u_2 \cdot \ldots \cdot u_n \cdot v_n$ is accepted by $\mathcal{T}$. Moreover, since $n \leq |\mathcal{D}ag(\mathcal{R})|$ and $|v_i| \leq |\mathcal{T}|$ for all $0 \leq i \leq n$, we have that the edit distance between u and $f(u)$ is at most $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$. This proves that there is a repair strategy f from $\mathcal{R}$ to $\mathcal{T}$ with worst-case cost at most $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$.

We now prove the contrapositive of the implication from *1)* to *2)*, namely, we assume that there is a path $\pi = C_1 \ldots C_k$ in $\mathcal{D}ag(\mathcal{R})$ that is not covered by any path π' in $\mathcal{D}ag^*(\mathcal{T})$ and from that we derive $\mathcal{D}ist(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})) = \omega$. First of all, we topologically sort the SCCs $C'_1, \ldots, C'_N$ of $\mathcal{T}$ in such a way that for every pair of indices $1 \leq j, j' \leq N$, if the states of $C'_{j'}$ are reachable from the states of C'_j, then $j \leq j'$ follows. We then associate with each pair of indices $1 \leq i \leq k$ and $1 \leq j \leq N$ a word $u_{i,j}$ defined by letting either $u_{i,j} = \varepsilon$ or $u_{i,j} \in \mathscr{L}(\mathcal{R}|C_i) \setminus \mathscr{L}(\mathcal{T}|C'_j)$, depending on whether $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|C'_j)$ or not. Since each word $u_{i,j}$ belongs to the language $\mathscr{L}(\mathcal{R}|C_i)$ and since all states in the SCC C_i along π are mutually reachable, we know that there exist some states $q_i, p_{i,1}, \ldots, p_{i,N} \in C_i$ and some words $u'_{i,1}, \ldots, u'_{i,N} \in \Delta^*$ such that $\mathcal{R}$ admits a run of the form

$$q_i \xrightarrow{u_{i,1}} p_{i,1} \xrightarrow{u'_{i,1}} q_i \ldots q_i \xrightarrow{u_{i,N}} p_{i,N} \xrightarrow{u'_{i,N}} q_i$$

namely, the word $v_i = u_{i,1} \cdot u'_{i,1} \cdot \ldots \cdot u_{i,N} \cdot u'_{i,N}$ is consumed by a run of $\mathcal{R}$ that is a loop inside the SCC C_i of π. From the above property, we also know that for every index $1 \leq i \leq k$ and every positive natural number n, the n-fold iteration v_i^n is accepted by the subautomaton $\mathcal{R}|C_i$. Similarly, since (i) all states in the first SCC C_1 along π are reachable from initial states, (ii) all states in the $i+1$-th SCC C_{i+1} are reachable from all states in the i-th SCC C_i along π, and (iii) all states in the last SCC C_n along π can reach some final states of $\mathcal{R}$, we know that there exist some words $v'_0, \ldots, v'_k$ such that, for every positive natural number n, the word $w_n = v'_0 \cdot v_1^n \cdot v'_1 \cdot v_2^n \cdot \ldots \cdot v_k^n \cdot v'_k$ is accepted by $\mathcal{R}$.
Below, we argue that the cost of repairing the family of words w_n into the regular language $\mathscr{L}(\mathcal{T})$ is unbounded. To do that, we consider a generic repair strategy $f : \mathscr{L}(\mathcal{R}) \to \mathscr{L}(\mathcal{T})$ and a family of corresponding successful runs ρ'_n of $\mathcal{T}$ on the words $f(u_n)$. For every positive natural number n, we denote by π'_n the path in $\mathcal{D}ag(\mathcal{T})$ that consists of the SCCs visited by

the run ρ'_n. Since $\mathcal{D}ag(\mathcal{T})$ contains only finitely many paths, we can assume, without loss of generality (i.e., by restricting to an infinite subset of the positive natural numbers), that all paths π'_n are the same. Accordingly, we denote these paths simply by $\pi'' = C''_1 \dots C''_h$, with $C''_1, \dots, C''_h$ being SCC of $\mathcal{T}$ among $C'_1 \dots C'_N$. We now establish the following claim:

> *For every path $\pi'' = C''_1 \dots C''_h$ in $\mathcal{D}ag^*(\mathcal{T})$, there exist some indices $1 = i_0 \le i_1 \le \dots \le i_h \le k$ such that (i) $\mathscr{L}(\mathcal{R}|C_{i_j}) \nsubseteq \mathscr{L}(\mathcal{T}|C''_j)$ for all $1 \le j \le h$ and (ii) $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|C''_j)$ for all $1 \le j \le h$ and all $i_{j-1} \le i < i_j$.*

We exploit an induction on h, namely, we assume that the claim holds for any path π'' of length h and we prove it for path of longer length $h+1$. The base case $h = 0$ is trivial. As for the inductive case, let $\pi'' = C''_1 \dots C''_h C''_{h+1}$ be a path in $\mathcal{D}ag^*(\mathcal{T})$. From the inductive hypothesis, we know that there exist some indices $1 = i_0 \le i_1 \le \dots \le i_h \le k$ such that (i) $\mathscr{L}(\mathcal{R}|C_{i_j}) \nsubseteq \mathscr{L}(\mathcal{T}|C''_j)$ for all $1 \le j \le h$ and (ii) $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|C''_j)$ for all $1 \le j \le h$ and all $i_{j-1} < i < i_j$. We distinguish between two cases: either $\mathscr{L}(\mathcal{R}|C_i) \nsubseteq \mathscr{L}(\mathcal{T}|C''_{h+1})$ for some $i_h \le i \le k$, or $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|C''_{h+1})$ for all $i_h < i \le k$. In the first case, we immediately obtain the claim by letting i_{h+1} be the first index $i_h \le i \le k$ such that $\mathscr{L}(\mathcal{R}|C_i) \nsubseteq \mathscr{L}(\mathcal{T}|C''_{h+1})$. In the second case, we obtain a contradiction when considering the path $\widetilde{\pi}'' = \widetilde{C}''_1 \dots \widetilde{C}''_k$ in $\mathcal{D}ag^*(\mathcal{T})$, where each SCC $\widetilde{C}''_l$, for $1 \le l \le k$, is defined to be either C''_j, where j is the unique index satisfying $i_{j-1} \le l < i_j$, if $l \le i_h$, or C''_{h+1} if $i_h < l \le k$. Indeed, by construction and by inductive hypothesis, we have $\mathscr{L}(\mathcal{R}|C_l) \subseteq \mathscr{L}(\mathcal{T}|\widetilde{C}''_l)$ for all $1 \le l \le i_h$ and, similarly, for all $i_h < l \le k$. This shows that the path $\pi = C_1 \dots C_k$ is covered by the path $\widetilde{\pi}'' = \widetilde{C}''_1 \dots \widetilde{C}''_k$, which is against the assumption on π. Therefore, we must conclude that only the first case can happen and hence the claim holds.

Turning to the proof of the main theorem, we know from the above claim that there exist some indices $1 \le i_1 \le \dots \le i_h \le k$ such that $\mathscr{L}(\mathcal{R}|C_{i_j}) \nsubseteq \mathscr{L}(\mathcal{T}|C''_j)$ for all $1 \le j \le h$. In particular, this shows that the word w_n defined previously contains n-occurrences of the substrings $u_{i_1,1}$, ..., $u_{i_h,h}$. We can thus write w_n as follows

$$w_n \;=\; \dots \; \underbrace{(u_{i_1,1} \dots u_{i_2,2} \dots\dots u_{i_h,h}) \;\dots\; (u_{i_1,1} \dots u_{i_2,2} \dots\dots u_{i_h,h})}_{n \text{ times}} \; \dots$$

Towards a conclusion, we recall that $\pi'' = C''_1 \dots C''_h$ is the path in $\mathcal{D}ag(\mathcal{T})$ that is visited by the successful run ρ_n on $f(w_n)$. By construction, each substring $u_{i_j,j}$ in w_n is not recognized by the subautomaton $\mathscr{L}(\mathcal{T}|C''_j)$. This shows that the word $f(w_n)$, which is accepted by $\mathcal{T}$ via the run ρ_n, must contain no occurrences of the substrings $u_{i_1,1} \dots u_{i_2,2} \dots\dots u_{i_h,h}$ (in this precise order). Therefore, the edit distance between w_n and $f(w_n)$ must be at least n and this implies that the repair strategy f has unbounded cost. ■

Theorem 3. *Given two DFA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

1) *there is a k-lookahead streaming strategy, for some $k \in \mathbb{N}$, that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded edit cost,*
2) *Eve has a winning strategy for the reachability game on $\mathcal{A}_{\mathcal{R},\mathcal{T}}$,*
3) *there is a 0-lookahead streaming strategy that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with worst-case aggregate cost at most $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$.*

Before turning to the proof of the above theorem, it is convenient to establish some preliminary lemmas, which can be reused also in other proofs (e.g., in the proof of Proposition 6). For the sake of brevity, given an NFA (resp., DFA) $\mathcal{A}$, a SCC C of it, and a state $q \in C$, we denote by $\mathcal{A}|_q C$ the NFA (resp., the DFA) obtained from the subautomaton $\mathcal{A}|C$ by letting q be the unique initial state (recall that the final states of $\mathcal{A}|C$ are all the states in C).

Lemma 24. *Let $\mathcal{R}$ be an NFA, let $\mathcal{T}$ be a DFA, and let C and C' be some SCCs of $\mathcal{R}$ and $\mathcal{T}$, respectively. We have that*

$$\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|C') \qquad \textit{iff} \qquad \exists\, q \in C,\, q' \in C'.\;\; \mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_{q'} C').$$

Proof: Let $\mathcal{R} = (\Sigma, Q, E, I, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', q'_0, F')$. We first prove the right-to-left implication. Suppose that $\mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_{q'} C')$ for some states $q \in C$ and $q' \in C'$. Let u be a word in $\mathscr{L}(\mathcal{R}|C)$. Since $q \in C$ and C is a SCC, we know that there is a word $u_0 \in \Sigma^*$ such that $u_0 \cdot u \in \mathscr{L}(\mathcal{R}|_q C)$. Since $\mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_{q'} C')$, the same word $u_0 \cdot u$ belongs also to the language $\mathscr{L}(\mathcal{T}|_{q'} C')$. In particular, this shows that $u \in \mathscr{L}(\mathcal{T}|C')$.

We now prove the contrapositive of the left-to-right implication, namely, we assume that $\mathscr{L}(\mathcal{R}|_q C) \nsubseteq \mathscr{L}(\mathcal{T}|_{q'} C')$ for all $q \in C$ and all $q' \in C'$ and we derive $\mathscr{L}(\mathcal{R}|C) \nsubseteq \mathscr{L}(\mathcal{T}|C')$. Let $C' = \{q'_1, \dots, q'_n\}$. We first prove, by exploiting an induction on $i \le n+1$, that there is a word u_i that belongs to $\mathscr{L}(\mathcal{R}|C)$ but not to $\mathscr{L}(\mathcal{T}|_{q'_j} C')$ for all indices $1 \le j < i$. The base case $i = 1$ is trivial. As for the inductive case, we assume that the claim holds for $i \le n$ and we prove it for $i+1$. Let u_i be the word obtained from the inductive hypothesis such that $u_i \in \mathscr{L}(\mathcal{R}|C) \setminus \mathscr{L}(\mathcal{R}|_{q'_j} C')$ for all $1 \le j < i$. Moreover, let r'_i be the state reached by $\mathcal{T}$ from q'_i after parsing the word u_i, that is, $r'_i = \delta'(q'_i, u_i)$. If $r'_i \notin C'$, then the claim follows trivially. It remains to consider the case $r'_i \in C'$. Let p_i and r_i be two states in C such that $p_i \xrightarrow{u_i} r_i$. We know from the original assumption that $\mathscr{L}(\mathcal{R}|r_i, C) \nsubseteq \mathscr{L}(\mathcal{T}|r'_i, C')$. In particular, we know that there is a word $v_i \in \mathscr{L}(\mathcal{R}|r_i, C) \setminus \mathscr{L}(\mathcal{T}|r'_i, C')$. This shows that the word $u_{i+1} = u_i \cdot v_i$ belongs to the language $\mathscr{L}(\mathcal{R}|C)$, but not to the language $\mathscr{L}(\mathcal{T}|_{q'_i} C')$. Finally, since C' is a SCC, it follows that u_{i+1} does not belong to any of the languages $\mathscr{L}(\mathcal{T}|_{q'_j} C')$ either, for all $1 \le j < i$, and this concludes the proof. ■

Lemma 25. *Let $\mathcal{R}$ and $\mathcal{T}$ be two DFA and let C and C' be some SCCs of $\mathcal{R}$ and $\mathcal{T}$, respectively. We have that*

$$\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|C') \qquad \textit{implies} \qquad \forall\, q \in C.\, \exists\, q' \in C'.\;\; \mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_{q'} C').$$

Proof: Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', q'_0, F')$ be two DFA such that $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|C')$, let C and C' be two SCC of $\mathcal{R}$ and $\mathcal{T}$, respectively, and let q be a state in C. We know from Lemma 24 that that there exist two states $p \in C$ and $p' \in C'$ such that $\mathscr{L}(\mathcal{R}|_p C) \subseteq \mathscr{L}(\mathcal{T}|_{p'} C')$. Moreover, since all states in C are reachable from each other, there is a word u such that $\delta(p, u) = q$. Since $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|C')$, we know that the state $q' = \delta(p', u)$ belongs to C'. Finally, for every word $v \in \mathscr{L}(\mathcal{R}|_q C)$, we have $u \cdot v \in \mathscr{L}(\mathcal{R}|_p C)$ and hence, since $\mathscr{L}(\mathcal{R}|_p C) \subseteq \mathscr{L}(\mathcal{T}|_{p'} C')$, $u \cdot v \in \mathscr{L}(\mathcal{T}|_{p'} C')$, whence $v \in \mathscr{L}(\mathcal{T}|_{q'} C')$. This shows that $\mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_{q'} C')$. ■

Proof of Theorem 3: Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', q'_0, F')$ be two DFA and let $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ be the arena obtained from $\mathcal{R}$ and $\mathcal{T}$ as described in Section IV. The implication from *3)* to *1)* is trivial. Below, we prove first the implication from *2)* to *3)* and then the implication from *1)* to *2)*.

Suppose that Eve has a winning strategy in the reachability game over $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. Since reachability games are positionally determined, Eve's strategy can be described by a partial function[1] g that maps a node of the form (D, C), with $C \in \mathcal{D}ag(\mathcal{R})$ and $D \in \mathcal{D}ag^*(\mathcal{T})$, to a successor $g(D, C) = (C, D')$ (if there is any) in the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. We can use Eve's winning strategy g to construct a sequential transducer $\mathcal{S}$ that implements a repair strategy of $\mathcal{R}$ into $\mathcal{T}$ having uniformly bounded aggregate cost. Intuitively, the transducer $\mathcal{S}$ works as follows. While parsing the input word u from the restriction language $\mathcal{R}$ and emitting a corresponding word v, the transducer $\mathcal{S}$ mimics, at the same time, the transitions of both $\mathcal{R}$ and $\mathcal{T}$. Each time the restriction automaton $\mathcal{R}$ exits the current SCC C and enters a new SCC C', a corresponding move $(C, D) \rightarrow (D, C')$ for Adam is identified; accordingly, on the basis of Eve's response $g(D, C') = (C', D')$ (recall that Eve's strategy was assumed to be winning), the transducer $\mathcal{S}$ outputs a suitable word that makes the target automaton $\mathcal{T}$ move from the SCC D to the SCC D' (the new state in D' can be determined using Lemma 25 since we have $\mathscr{L}(\mathcal{R}|C') \subseteq \mathscr{L}(\mathcal{R}|D')$).
The formal definition of the transducer $\mathcal{S}$ is a bit more technical due to the treatment of some special cases. First of all, we can assume, without loss of generality, that the initial state q_0 of $\mathcal{R}$ has no entering transitions (we can always enforce this condition by introducing duplicate states). Then, we let C_0 be the SCC of q_0 in $\mathcal{R}$, D_0 be the SCC of q'_0 in $\mathcal{T}$, $(C_0, D_1) = g(D_0, C_0)$ be target node in the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ for the first move of the player Eve (recall that g is the winning strategy of Eve), and q'_1 be some arbitrary state in D_1 such that $\mathscr{L}(\mathcal{R}|_{q_0} C_0) \subseteq \mathscr{L}(\mathcal{T}|_{q'_1} D_1)$ (note that $\mathscr{L}(\mathcal{R}|C_0) \subseteq \mathscr{L}(\mathcal{R}|D_1)$ and hence, by Lemma 25, such a state q'_1 must exist). Accordingly, we define v_0 to be the shortest word such that $\delta'(q'_0, v_0) = q'_1$ (note that since (D_0, D_1) is an edge in $\mathcal{D}ag^*(\mathcal{T})$, the state q'_1 is reachable from the state q'_0). Intuitively, the word v_0 is the first prefix emitted by the transducer $\mathcal{S}$ at the beginning of the computation (it should be now clear why we assumed that the initial state q_0 of $\mathcal{R}$ has no entering transitions). We are finally ready to define the transducer $\mathcal{S} = (\Sigma, \Delta, Q'', \delta'', q''_0, \Omega)$:

- $Q'' = Q \times Q'$;
- for every state $p'' = (p, p') \in Q''$ and every symbol $a \in \Sigma$, $\delta''\big(p'', a\big) = \big(v, q''\big)$, where $q'' = (\delta(p, a), \delta'(p', v))$ and v is the word over Δ defined as follows
  1) if both states p and $\delta(p, a)$ belong to the same SCC of $\mathcal{R}$, then we let v be either the input symbol a or the word $v_0 \cdot a$, depending on whether $p \neq q_0$ or $p = q_0$;
  2) otherwise, if the states p and $\delta(p, a)$ belong to different SCCs, we denote by C' the SCC of the state $\delta(p, a)$ in $\mathcal{R}$, by D the SCC of the state p' in $\mathcal{T}$, by (C', D') the response $g(D, C')$ given by Eve's winning strategy, by q' some arbitrary state in D' such that $\mathscr{L}(\mathcal{R}|_{p'} C') \subseteq \mathscr{L}(\mathcal{T}|_{q'} D')$ (we know from the containment $\mathscr{L}(\mathcal{R}|C') \subseteq \mathscr{L}(\mathcal{R}|D')$ and from Lemma 25 that such a state q' exists), and by v' the shortest word such that $\delta'(p', v') = q'$ (note that since (D, D') is an edge in $\mathcal{D}ag^*(\mathcal{T})$, the state q' is reachable from the state p'); accordingly, we let v be either the word v' or the word $v_0 \cdot v'$, depending on whether $p \neq q_0$ or $p = q_0$;
- $q''_0 = (q_0, q'_0)$;
- for every state $p'' = (p, p') \in Q''$, $\Omega(p'')$ is the shortest word $v \in \Delta^*$ such that $\delta(p', v) \in F'$ (note that the existence of such a word v is guaranteed by the assumption that every state of $\mathcal{T}$ can reach some final state).

[1]The reason for the strategy function g to be partial is that some positions in the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ may have no successors.

Using arguments similar to those used in the proof of Theorem 2 (e.g., by associating with each successful run of $\mathcal{R}$ a corresponding path π in $\mathcal{D}ag(\mathcal{R})$ and a covering path π' in $\mathcal{D}ag^*(\mathcal{T})$ induced by Eve's winning strategy), one can show that the transducer $\mathcal{S}$ maps any word $u \in \mathscr{L}(\mathcal{R})$ to a word $\mathcal{S}(u) \in \mathscr{L}(\mathcal{T})$. Moreover, by construction, the output produced at each transition of $\mathcal{S}$ can be different from the input symbol only if $\mathcal{S}$ is at the beginning of the computation, at the end of the computation, or if the current SCC of the automaton $\mathcal{R}$ has just changed; in each of these cases, the length of the produced output does not exceed the number of states in $\mathcal{T}$. This shows that the the aggregate cost of $\mathcal{S}$ on input $u \in \mathscr{L}(\mathcal{R})$ is at most $(1+|\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$. We thus conclude that there there is a 0-lookahead streaming repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ having worst-case aggregate cost at most $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$.

We now prove the implication from *1)* to *2)*, namely, we assume that $\mathcal{S} = (\Sigma, \Delta, Q'', \delta'', q_0'', \Omega)$ is a transducer with k-lookahead that implements a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with cost uniformly bounded by a natural number N and we construct from this a winning strategy for Eve. More precisely, we shall specify Eve's moves $g(D, C)$ on those nodes (D, C) in $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, with $C \in \mathcal{D}ag(\mathcal{R})$ and $D \in \mathcal{D}ag^*(\mathcal{T})$, for which there exist some words $u_C, u'_C \in \Sigma^*$ and $v_D \in \Delta^*$ such that

i) $\delta(q_0, u_C \cdot u'_C) \in C$,

ii) $\delta'(q'_0, v_D) \in D$,

iii) $\delta''(q''_0, u_C) = (v_D, q'')$ for some $q'' \in Q''$.

We call these nodes *reachable*. On the remaining (unreachable) nodes, we let the function g be unspecified. As a preliminary remark, we observe that from the definition of the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ and from the above properties, the domain of the function g is closed under the ancestor relation, namely, if $g(D, C)$ is defined and (D', C') is an ancestor of (D, C) in $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, then $g(D', C')$ is defined as well. Below, we define the moves $g(D, C)$ by exploiting an induction on the distance of the SCC C from the root of $\mathcal{D}ag(\mathcal{R})$, which is the SCC of the initial state q_0 of $\mathcal{R}$.

Let us consider a reachable node (D, C) in the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. Since (D, C) is reachable, we know that there exist some words $u_C, u'_C \in \Sigma^*$ and $v_D \in \Delta^*$ such that (i) $\delta(q_0, u_C \cdot u'_C) \in C$, (ii) $\delta'(q'_0, v_D) \in D$, and (iii) $\delta''(q''_0, u_C) = (v_D, q'')$ for some $q'' \in Q''$. For the sake of brevity, we let $p = \delta(q_0, u_C \cdot u'_C)$. We claim that there is a SCC D' that is a successor of D in $\mathcal{D}ag^*(\mathcal{T})$ (possibly $D' = D$) such that $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D')$. Indeed, suppose, by way of contradiction that this is not the case. Let $D_1, \ldots, D_h$ be all the the descendants of D, topologically ordered according to their accessibility relation (hence $D_1 = D$). Using arguments similar to those used in the proof of Theorem 2, we can recursively construct a sequence of words $u_0, u_1, u'_1, \ldots, u_h, u'_h$ over Σ such that

i) $p \xrightarrow{u_i \cdot u'_i} p$ is a run of $\mathcal{R}$ on $u_i \cdot u'_i$ for all $1 \leq i \leq h$,

ii) $u_i \notin \mathscr{L}(\mathcal{T}|D_i)$ for all $1 \leq i \leq h$.

In particular, the first property implies that the word

$$u_C \cdot u'_C \cdot (u_1 \cdot u'_1)^{N+1} \cdot \ldots \cdot (u_h \cdot u'_h)^{N+1}$$

is a prefix of some word u in the language $\mathscr{L}(\mathcal{R})$ (recall the assumption that all states in $\mathcal{R}$ can reach some final states). Moreover, since u contains $N + 1$ occurrences of the subwords $u_1, \ldots, u_h$ and since $\mathcal{S}$ implements a repair strategy with cost uniformly bounded by N, we have that the words $u_1, \ldots, u_h$ must occur at least once, and in this particular order, as subwords of $\mathcal{S}(u)$ (observe that having a transducer with k-lookahead does not help here). However, since v_D is a prefix of $\mathcal{S}(u)$, $\delta(q'_0, v_D) \in D$, and $u_i \notin \mathscr{L}(\mathcal{T}|D_i)$ for all $1 \leq i \leq h$, we have that $\mathcal{S}(u)$ can not belong to the target language $\mathscr{L}(\mathcal{T})$, which is against the assumption that $\mathcal{S}$ implements a repair strategy of $\mathcal{R}$ into $\mathcal{T}$. We must conclude that there is a SCC D' that is a successor of D in $\mathcal{D}ag^*(\mathcal{T})$ and that satisfies $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D')$. Accordingly, we define Eve's move on node (D, C) to be $g(D, C) = (C, D')$ (note that this is a valid move in the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$).

Turning to the proof of the main theorem, we argue that the above defined strategy g for Eve is winning. This is equivalent to proving that for every partial play of the form

$$(D_0, C_0) \xrightarrow{\text{Eve}} (C_0, D_1) \xrightarrow{\text{Adam}} \ldots \xrightarrow{\text{Adam}} (D_n, C_n)$$

that follows Eve's strategy g, Eve is able to respond with an appropriate move, namely, g is defined on the node (D_n, C_n). In fact, to prove this property it is sufficient to show that the node (D_n, C_n) is reachable and this can be easily verified by exploiting an induction on n and the definition of the strategy g given above. We thus conclude that Eve has a winning strategy for the reachability game on $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. ■

B. Proofs for Section V (Complexity results in the non-streaming case)

Theorem 5. *The bounded repair problem in the non-streaming case, where the restriction language is represented by an NFA and the target language is represented by a DFA, is in coNP and it is coNP-hard already for languages represented by DFA.*

We first establish the following complexity result for the coverability problem:

Lemma 26. *Given two NFA $\mathcal{R}$ and $\mathcal{T}$ and a path π in $\mathcal{D}ag(\mathcal{R})$, the problem of deciding whether π is covered by some path π' in $\mathcal{D}ag(\mathcal{T})$ is in PTIME with an oracle for deciding containment between the languages of the form $\mathscr{L}(\mathcal{R}|\pi(i))$ and $\mathscr{L}(\mathcal{T}|\pi'(i))$.*

Proof: Let $\mathcal{R}$ and $\mathcal{T}$ be two NFA and let π be a path in $\mathcal{D}ag(\mathcal{R})$. The basic idea for checking whether the path π is covered by some path π' in $\mathcal{D}ag(\mathcal{T})$ is to perform a sort of online subset construction on $\mathcal{D}ag^*(\mathcal{T})$, that is, to incrementally process larger and larger prefixes of π while keeping the frontier of the paths in $\mathcal{D}ag(\mathcal{T})$ that cover these prefixes. Algorithm A.1 below describes the pseudo-code of an algorithm that implements this idea.

Algorithm A.1: PATHCOVERABILITY($\mathcal{R}$, $\mathcal{T}$, π)

```
let $\pi = C_1 \dots C_k$
let $\mathcal{D}ag^*(\mathcal{T}) = (V', E')$
$F \leftarrow \emptyset$
for all $C \in V'$
    do if CHECKCONTAINMENT($\mathcal{R}$, $\mathcal{T}$, $C_1$, $C$)
         then $F \leftarrow F \cup \{C\}$
for $i \leftarrow 2$ to $k$
    do $F' \leftarrow F$
       $F \leftarrow \emptyset$
       for all $C' \in F'$ and $(C', C) \in E'$
           do if CHECKCONTAINMENT($\mathcal{R}$, $\mathcal{T}$, $C_i$, $C$)
                then $F \leftarrow F \cup \{C\}$
return $(F \neq \emptyset)$
```

Note that the pseudo-code uses CHECKCONTAINMENT($\mathcal{R}, \mathcal{T}, C_i, C$) as an oracle for deciding containment between the languages $\mathscr{L}(\mathcal{R}|C_i)$ and $\mathscr{L}(\mathcal{T}|C)$. The proof of the correctness of the algorithm is straightforward and it is based on the following invariant: at each iteration of the loop on i, the set F contains a SCC C of $\mathcal{T}$ iff there is a path $\pi' = C'_1 \dots C'_i$ in $\mathcal{D}ag^*(\mathcal{T})$, with $C'_i = C$, that covers $\pi_i = C_1 \dots C_i$. Clearly the described procedure runs in polynomial time with respect to the size of the input NFA $\mathcal{R}$ and $\mathcal{T}$. ■

Proof of Theorem 5: Let $\mathcal{R}$ be an NFA and $\mathcal{T}$ be a DFA. In view of the characterization result of Theorem 2, deciding whether there is a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost amounts at first guessing universally a path π in $\mathcal{D}ag(\mathcal{R})$ (this can be done in coNP) and then checking whether π is covered by some path in $\mathcal{D}ag(\mathcal{T})$ (by Lemma 26, this can be done in PTIME using an oracle for deciding the containment of languages recognized by SCCs of $\mathcal{R}$ and $\mathcal{T}$). Moreover, recall that Lemma 24 reduces the containment problem between the language recognized by a SCC of the NFA $\mathcal{R}$ and the language recognized by a SCC of the DFA $\mathcal{T}$ to containment problem between a non-deterministic subautomaton of $\mathcal{R}$ and a deterministic subautomaton of $\mathcal{T}$, which is known to be in PTIME (indeed, given an NFA $\mathcal{A}$ and a DFA $\mathcal{B}$, we have $\mathscr{L}(\mathcal{A}) \subseteq \mathscr{L}(\mathcal{B})$ iff $\mathscr{L}(\mathcal{A}) \cap \mathscr{L}(\mathcal{B}^{\complement}) = \emptyset$, where $\mathcal{B}^{\complement}$ denotes the DFA that recognizes the complement of $\mathscr{L}(\mathcal{B})$). Putting all together, we have that the bounded repair problem in the non-streaming case, where the restriction language is given by an NFA and the target language is given by a DFA, is in coNP.

As for the hardness result, we reduce the validity problem of propositional formulas in disjunctive normal (which is known to be coNP-hard) to the bounded repair problem for DFA in the nonstreaming setting. We fix a set $X = \{x_1, \dots, x_k\}$ of propositional variables and we denote by $L = \{x_1, \dots, x_k\} \cup \{\neg x_1, \dots, \neg x_k\}$ the corresponding set of literals. For the sake of brevity, we identify $\neg\neg x_i$ with x_i. A valuation for the variables in X can be viewed as a subset V of L such that, for every literal $l \in L$, we have $l \in V$ iff $\neg l \in V$. Let use consider a formula in disjunctive normal form

$$\varphi = \bigvee_{1 \le i \le m} \bigwedge_{1 \le j \le h_i} l_{i,j}$$

with $l_{i,j} \in L$ for every pair of indices $1 \leq i \leq m$ and $1 \leq j \leq h_i$. Below, we describe suitable restriction and target languages R and T such that R can be repaired into T with uniformly bounded cost iff φ is valid, namely, if for every valuation V, there is an index $1 \leq i \leq m$ such that $l_{i,1}, \ldots, l_{i,h_i} \in V$.

We define the restriction language over the alphabet $\Sigma = L$ to be

$$R \;=\; \big(\{x_1\}^* \cup \{\neg x_1\}^*\big) \cdot \ldots \cdot \big(\{x_k\}^* \cup \{\neg x_k\}^*\big)$$

Note that it is easy to construct a DFA $\mathcal{R}$ that recognizes R and that has size linear in the number of variables used by φ. Similarly, we define the target language over the alphabet $\Delta = \{a_1, \ldots, a_m\} \cup L$ to be

$$T \;=\; \bigcup_{1 \leq i \leq m} \{a_i\} \cdot \big(L \setminus \{\neg l_{i,1},\, \ldots,\, \neg l_{i,h_i}\}\big)^*.$$

Again, it is easy to construct a DFA $\mathcal{T}$ that recognizes T and that has size linear in the number of clauses of φ and in the number of its variables.

We verify that R can be repaired into T with uniformly bounded cost iff φ is valid. As for the right-to-left implication, suppose that φ is valid and let u be a word in R. Clearly, u is a word of the form $(l_1 \ldots l_1)(l_2 \ldots l_2) \ldots (l_k \ldots l_k)$, with $l_i \in \{x_i, \neg x_i\}$ for all $1 \leq i \leq k$. Such a word encodes the valuation $V_u = \{l_1, l_2, \ldots, l_k\}$. Moreover, since φ is valid, there is an index $1 \leq i \leq m$ such that $l_{i,1}, \ldots, l_{i,h_i} \in V$. The repair strategy f of R into T could then map the word u to the word $f(u) = i \cdot u$, which clearly belongs to T. As for the converse implication, we assume that φ is not valid. This means that there is a valuation $V = \{l_1, l_2, \ldots, l_k\}$, with $l_i \in \{x_1, \neg x_i\}$ for all $1 \leq i \leq k$, such that for every index $1 \leq i \leq m$, there is an index $1 \leq j \leq h_i$ satisfying $l_{i,j} \notin V$. We thus consider the family of words $u_n = (l_1)^n \cdot (l_2)^n \ldots \cdot (l_k)^n)$. From previous arguments, we know that, for every $1 \leq i \leq m$, there is $1 \leq j \leq h_i$ such that the edit distance between the subword $(\neg l_{i,j})^n$ of u_n and any word in the sublanguage $T_i = \{a_i\} \cdot \big(L \setminus \{\neg l_{i,1},\, \ldots,\, \neg l_{i,h_i}\}\big)^*$ of T is at least n. This shows that any repair strategy of R into T has unbounded cost. ■

Proposition 6. *Let R be a fixed restriction language. The problem of deciding, given a DFA $\mathcal{T}$, whether there is a non-streaming repair strategy of R into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost is in PTIME.*

Proof: The proof is similar to the part of the proof of Theorem 5 regarding the coNP upper bound. Let $\mathcal{R}$ be a fixed DFA recognizing the restriction language R and let $\mathcal{T}$ be a given DFA recognizing the target language T. From Theorem 2, deciding whether there is a non-streaming repair strategy of R into T amounts at checking that every path π in $\mathcal{D}ag(\mathcal{R})$ is covered by some path in $\mathcal{D}ag(\mathcal{T})$. In virtue of Lemma 26 and Lemma 24, coverability of a given path in $\mathcal{D}ag(\mathcal{R})$ by paths in $\mathcal{D}ag^*(\mathcal{T})$ can be decided in PTIME. Since the number of paths in $\mathcal{D}ag(\mathcal{R})$ is fixed, this shows that the problem in decidable in polynomial time. ■

Proposition 7. *Let T be a fixed target language. The problem of deciding, given an NFA $\mathcal{R}$, whether there is a non-streaming repair strategy of $\mathscr{L}(R)$ into T with uniformly bounded cost is in PTIME.*

Proof: The polynomial-time solution to the bounded repair problem under a fixed target language $T = \mathscr{L}(\mathcal{T})$ uses the characterization of Theorem 2 (and hence it is similar to that of Theorem 5). However, instead of guessing a path π in $\mathcal{D}ag(\mathcal{R})$ and then checking whether π is covered by some path π' in $\mathcal{D}ag^*(\mathcal{T})$, we directly compute (a set representing) all instances of the coverability relation. More precisely, we compute the set P of all pairs of the form (C, F), where C is a SCC of the NFA $\mathcal{R}$ and $F = \{D_1, \ldots, D_n\}$ is a set of SCCs of the DFA $\mathcal{T}$, for which there is a path π in $\mathcal{D}ag(\mathcal{R})$ that ends in C and that is covered by the paths $\pi'_1, \ldots, \pi'_n$ in $\mathcal{D}ag^*(\mathcal{T})$, with each π'_i ending in D_i (of course we have repeated occurrences of the same component among $D_1, \ldots, D_n$). We call such a path π a *witness* of the pair (C, F). Clearly, if $F \neq \emptyset$ for all pairs $(C, F) \in P$, then we know that every path π in $\mathcal{D}ag(\mathcal{R})$ is covered by some path in $\mathcal{D}ag^*(\mathcal{T})$ (and similarly for the converse implication). The elements (C, F) of the set P can be recursively computed by exploiting an induction on the length of the witnessing paths. Algorithm A.2 below implements such an idea.

Algorithm A.2: ALLPATHSCOVERABILITY($\mathcal{R}, \mathcal{T}$)

```
let $\mathcal{D}ag(\mathcal{R}) = (V, E)$
let $\mathcal{D}ag^*(\mathcal{T}) = (V', E')$
$P \leftarrow \emptyset$
for all $C \in V$
  do
    $F \leftarrow \emptyset$
    for all $D \in V'$
      do
        if CHECKCONTAINMENT($\mathcal{R}$, $\mathcal{T}$, $C$, $D$)
          then $F \leftarrow F \cup \{D\}$
    $P \leftarrow P \cup \{(C, F)\}$
for $k \leftarrow 2$ to $|V|$
  do
    for all $(C, F) \in P$ and $(C, C') \in E$
      do
        $F' \leftarrow \emptyset$
        for all $D \in F$ and $(D, D') \in E'$
          do
            if CHECKCONTAINMENT($\mathcal{R}$, $\mathcal{T}$, $C'$, $D'$)
              then $F' \leftarrow F' \cup \{D'\}$
        $P \leftarrow P \cup \{(D', F')\}$
return $(\forall\, (C, F) \in P.\ F \neq \emptyset)$
```

The proof that the algorithm is correct, namely, that it returns `true` iff every path π in $\mathcal{D}ag(\mathcal{R})$ is covered by some path in $\mathcal{D}ag^*(\mathcal{T})$, relies on the following invariant: at each iteration of the loop on k, the set P contains all pairs (C, F) that are witnessed by a path π in $\mathcal{D}ag(\mathcal{R})$ of length at most k. Moreover, we observe that every instruction used in the pseudo-code of the algorithm (including the calls to the subroutine CHECKCONTAINMENT($\mathcal{R}, \mathcal{T}, C, D$)) requires time polynomial in the size of the arguments (recall, for instance, Lemma 24). Finally, since the set P has size at most $|\mathcal{R}| \times 2^{|\mathcal{T}|}$ and $\mathcal{T}$ is fixed, we can conclude that the algorithm runs in polynomial time with respect to the size of $\mathcal{R}$. ■

Theorem 8. *The bounded repair problem in the non-streaming case, where the restriction and target languages are represented by LTL formulas, is coNEXPTIME-complete.*

Proof: We already explained the complexity upper bound in Section V so we focus here on the lower bound. To prove coNEXPTIME-hardness, we reduce the problem of deciding the non-existence of a tiling of an exponential square grid (which is known to be coNEXPTIME-hard [8] to the problem of deciding the existence of a repair strategy of uniformly bounded cost between two regular languages defined by suitable LTL formulas. We briefly introduce some definitions for the tiling problem. A tiling instance, is a tuple $I = (n, S, H, V, t_\perp, t_\top)$, where S is a finite set of tiles, $H, V \subseteq S \times S$ are the relations for the horizontal and vertical constraints between tiles, 2^n is the length of the edge of the square grid $\mathcal{G}_n = [0, 2^n - 1] \times [0, 2^n - 1]$ to be tiled (n is represented in unary notation and thus within space $|n|$), and $t_\perp$ and $t_\top$ are the tiles that must appear at the bottom and top rows of the grid. A tiling (for the instance I) is a mapping g from the pairs $(i, j) \in \mathcal{G}_n$ to the tiles in S such that $g(0, j) = t_\perp$ and $g(2^n - 1, j) = t_\top$ for all $0 \leq j < 2^n$. We say that the tiling g satisfies the constraints of I if, for every pair of indices $0 \leq i, j < 2^n$, with $j > 0$ (resp., $i > 0$), we have $\big(g(i, j-1), g(i, j)\big) \in H$ (resp., $\big(g(i-1, j), g(i, j)\big) \in V$). The (exponential square grid) tiling problem is the problem of deciding, given a tiling instance $I = (n, S, H, V, t_\perp, t_\top)$, whether there is a tiling g that satisfies the constraints in I. It is known (see, for instance, [8]) that this problem is NEXPTIME-complete. Below, we fix a tiling instance $I = (n, S, H, V, t_\perp, t_\top)$ and we construct suitable LTL formulas ϕ and ψ that have size polynomial in $|I|$ and that define two regular languages R and T such that R can be repaired into T with uniformly bounded cost iff there is *no* tiling of $\mathcal{G}_n$ that satisfies the constraints in I. This would imply that the bounded repair problem for LTL formulas is coNEXPTIME-complete.

We start by defining the formula ϕ for the restriction language R. The idea is to define the restriction language R as the set of all (redundant) encodings of the tilings of $\mathcal{G}_n$. For the sake of brevity, we let $N = 2^n$. We let the restriction alphabet be $\Sigma = \{0, 1\#\} \uplus S$ (using slightly more encoding, one can even assume that the restriction alphabet contains only two symbols) and we define the restriction language R to be the union of the languages

$$L_g \;=\; \{u_{0,0}\}^+ \cdot \ldots \cdot \{u_{0,N-1}\}^+ \cdot \ldots\ldots \cdot \{u_{N-1,0}\}^+ \cdot \ldots \cdot \{u_{N-1,N-1}\}^+.$$

for all tiling functions $g : \mathcal{G}_n \;\rightarrow\; S$, where $u_{i,j}$ is the word $\langle i, j\rangle g(i, j)\#$ and $\langle i, j\rangle$ is the $2n$-character word obtained from the juxtaposition of the binary encodings of the coordinates i and j (e.g., $\langle 0, 0\rangle = 00\ldots0\,00\ldots0$, $\langle 1, 0\rangle = 10\ldots0\,00\ldots0$,

$\langle N, 0\rangle = 11\ldots1\,00\ldots0$, etc.). We now show how to define the language R using an LTL formula ϕ of size polynomial in $|I|$ (i.e., in n and $|S|$). The formula ϕ will be a conjunction of the form

$$\phi \;=\; \mathbf{G}\big((\phi_{\text{block}} \;\wedge\; \mathbf{X}^{2n+2}\,\mathtt{true}) \;\to\; (\mathbf{X}^{2n+2}\phi_{\text{block}} \;\wedge\; \phi_{\text{succ}} \;\wedge\; \phi_{\text{corr}})\big) \;\;\wedge\;\; \phi_{\text{init}} \;\;\wedge\;\; \phi_{\text{final}}$$

where ϕ_{block}, ϕ_{succ}, ϕ_{corr}, ϕ_{init}, and ϕ_{final} are suitable formulas to be defined below. The formula ϕ_{block} guarantees that the next $2n+2$ symbols (hereafter called block) form word of the form $\langle i,j\rangle t\#$, for some $0 \le i,j < N$ and some $t \in S$. We let

$$\phi_{\text{block}} \;=\; \bigwedge_{0\le k<2n} \mathbf{X}^k(0 \,\vee\, 1) \;\;\wedge\;\; \bigvee_{t\in S} \mathbf{X}^{2n}t \;\;\wedge\;\; \mathbf{X}^{2n+1}\#.$$

The formula ϕ_{succ} enforces a matching relation between contiguous blocks, namely, it checks that for the next 2 blocks $\langle i,j\rangle t\#$ and $\langle i',j'\rangle t'\#$, we have either $i'=i$ and $j'=j$, or $i'=i$ and $j'=j+1$, or $i'=i+1$, $j=N-1$, and $j'=0$. This can be done by defining

$$\begin{aligned}\phi_{\text{succ}} \;\;=\;\; & \bigwedge_{0\le k<2n}\Big(\mathbf{X}^k 0 \;\leftrightarrow\; \mathbf{X}^{2n+2+k}0\Big) \;\vee \\ & \bigvee_{0\le k<2n}\Big(\bigwedge_{0\le h<k}\mathbf{X}^h 1 \;\wedge\; \mathbf{X}^{2n+2+h}0\Big) \;\wedge\; \Big(\mathbf{X}^k 0 \;\wedge\; \mathbf{X}^{2n+2+k}1\Big) \;\wedge\; \Big(\bigwedge_{k<h<2n}\mathbf{X}^h 0 \;\leftrightarrow\; \mathbf{X}^{2n+2+h}0\Big).\end{aligned}$$

The formula ϕ_{corr} guarantees that the tiles specified in two contiguous blocks with the same coordinates i and j agree, namely, that for the next 2 blocks $\langle i,j\rangle t\#$ and $\langle i',j'\rangle t'\#$, if $i=i'$ and $j=j'$, then $t=t'$. This can be done by letting

$$\phi_{\text{corr}} \;=\; \bigwedge_{0\le k<2n}\Big(\mathbf{X}^k 0 \;\leftrightarrow\; \mathbf{X}^{2n+2+k}0\Big) \;\to\; \bigwedge_{t\in S}\Big(\mathbf{X}^{2n}t \;\leftrightarrow\; \mathbf{X}^{4n+2}t\Big).$$

Finally, the remaining two formulas $\phi_{\text{init}} = \big(\bigwedge_{0\le k<2n}\mathbf{X}^k 0\big) \;\wedge\; \mathbf{X}^{2n}t_\perp \;\wedge\; \mathbf{X}^{2n+1}\#$ and $\phi_{\text{final}} = \mathbf{F}\big(\big(\bigwedge_{0\le k<2n}\mathbf{X}^k 1\big) \;\wedge\; \mathbf{X}^{2n}t_\top \;\wedge\; \mathbf{X}^{2n+1}\# \;\wedge\; \neg\mathbf{X}^{2n+2}\big)$ enforce the usual boundary conditions, namely, they require that the first $2n+2$ symbols of the word form a block of the form $\langle 0,0\rangle t_\perp\#$ and, similarly, the last $2n+2$ symbols of the word form a block of the form $\langle N-1,N-1\rangle t_\top\#$. It is routine to verify that the language defined by the formula ϕ is precisely the restriction language R.

We now turn to the definition of the formula ψ for the target language T. The idea is that every word in the target language must be obtained from a word in the restriction language (i.e., from the encoding of a tiling function) by adding a "certificate" that makes it easy to verify that the encoded tiling function does not satisfy the horizontal and vertical constraints of I. More precisely, the certificate will be defined as a decoration of a block $\langle i,j\rangle t\#$ (e.g., by means of underlined symbols) that makes it distinguishable from all other blocks in the word; in such way, a suitable LTL formula (of small size) can easily check whether the tiles at the positions (i,j), $(i-1,j)$, and $(i,j-1)$ are consistent or not with the horizontal and vertical constraints of I. The alphabet of the target language is $\Delta = \Sigma \uplus \{\underline{\#}\}$, where $\underline{\#}$ is a new symbol not in Σ. Given a word v over Δ, we denote by $v[\underline{\#}/\#]$ the word over Σ obtained from v by replacing every occurrence of the symbol $\underline{\#}$ with $\#$. Formally, the target language T is defined as the set of all and only the words of the form

$$v \;=\; \ldots\; \langle i-1,j\rangle t\# \;\ldots\; \langle i,j-1\rangle t'\# \; \langle i,j\rangle t''\underline{\#} \;\ldots$$

such that (i) $v[\underline{\#}/\#]$ belongs to the restriction language R and (ii) $(t',t'') \notin H$ or $(t,t'') \notin V$. It is easy to verify that the target language T is defined by the following LTL formula of size polynomial in n and $|S|$:

$$\psi \;=\; \phi\big[\#/(\# \,\vee\, \underline{\#})\big] \;\wedge\; \psi_{\text{mark}} \;\wedge\; \big(\psi_{\text{check-H}} \,\vee\, \psi_{\text{check-V}}\big)$$

where

- $\phi\big[\#/(\# \vee \underline{\#})\big]$ is obtained from ϕ (i.e., the formula that defines the restriction language R) by replacing every occurrence of the propositional variable $\#$ by the disjunction $\# \,\vee\, \underline{\#}$;
- ψ_{mark} is the formula $(\neg\underline{\#})\;\mathbf{U}\;(\underline{\#} \;\wedge\; \mathbf{G}\neg\underline{\#})$, which enforces precisely one occurrence of the underlined symbol $\underline{\#}$;
- $\psi_{\text{check-H}}$ is the formula $\mathbf{F}\big(\bigwedge_{0\le k<n}(\mathbf{X}^k 0 \;\leftrightarrow\; \mathbf{X}^{2n+2+k}0) \;\wedge\; \neg\bigwedge_{n\le k<2n}(\mathbf{X}^k 0 \;\leftrightarrow\; \mathbf{X}^{2n+2+k}0) \;\wedge\; \bigvee_{(t',t'')\notin H}(\mathbf{X}^{2n+1}t' \;\wedge\; \mathbf{X}^{4n+3}t'')\big)$, which verifies that the horizontal constraints are violated by the tiles associated with the underlined block and with its predecessor along the same row, but on a different column;
- $\psi_{\text{check-V}}$ is a formula that verifies that the vertical constraints are violated by the tiles associated with the underlined block and with some previous block along the same column, but on a different row; such a formula can be written as

$$\begin{aligned}\mathbf{F}\Big(& \bigwedge_{0\le k<n}\big(\mathbf{X}^k 0 \leftrightarrow \mathbf{F}(\mathbf{X}^{2n+1}\underline{\#} \;\wedge\; \mathbf{X}^{2n+2+k}0)\big) \;\wedge \\ & \neg\bigwedge_{n\le k<2n}\big(\mathbf{X}^k 0 \leftrightarrow \mathbf{F}(\mathbf{X}^{2n+1}\underline{\#} \;\wedge\; \mathbf{X}^{2n+2+k}0)\big) \;\wedge \\ & \bigvee_{(t',t'')\notin H}\big(\mathbf{X}^{2n+1}t' \wedge \mathbf{F}(\mathbf{X}^{2n+1}\underline{\#} \;\wedge\; \mathbf{X}^{4n+3}t'')\big)\Big).\end{aligned}$$

It remains to prove that R can be repaired into T with uniformly bounded cost iff every tiling of $\mathcal{G}_n$ violates the constraints in I. As for the right-to-left direction, we assume that every tiling of $\mathcal{G}_n$ violates the constraints in I. This means that for every tiling g, there exist two indices $0 \leq i,j < N$ such that $\big(g(i,j-1), g(i,j)\big) \notin H$ or $\big(g(i-1,j), g(i,j)\big) \notin V$. By exploiting this property, one can repair any word $u \in R$, which encodes a tiling g, into a word $v \in T$, by simply replacing a distinguished occurrence of $\#$ with $\underline{\#}$. This shows that there is a repair strategy with worst-case cost 1. As for the converse direction, we assume that there is a valid tiling g of $\mathcal{G}_n$ and we consider the family of words

$$u^{(n)} \;=\; u^n_{0,0} \cdot \ldots \cdot u^n_{0,N-1} \cdot \;\ldots\ldots\; \cdot u^n_{N-1,0} \cdot \ldots \cdot u^n_{N-1,N-1}$$

where, for every $0 \leq i,j < N$, $u_{i,j}$ is the block $\langle i,j \rangle \, g(i,j) \, \#$. By exploiting the fact that all blocks in $u^{(n)}$ satisfy the horizontal and the vertical constraints, one can easily verify that the edit distance between the word $u^{(n)}$ and any word in the target language T is at least n, for all natural numbers n. This shows that any repair strategy of R into T has unbounded cost. ■

Theorem 9. *The bounded repair problem in the non-streaming case, where the restriction language R is represented by an NFA and the target language T is represented by an LTL formula, is in PSPACE.*

Proof: Let us fix an NFA $\mathcal{R}$ recognizing a restriction language R and an LTL formula ψ defining a target language T. In a way similar to the proof of the coNEXPTIME complexity bound for the LTL-vs-LTL bounded repair problem (see Theorem 8), we reduce the problem to deciding whether there is a non-streaming repair strategy of $\overleftarrow{R}$ into $\overleftarrow{T}$, where $\overleftarrow{R}$ and $\overleftarrow{T}$ are the reversal languages of R and T, respectively. Clearly, from the NFA $\mathcal{R}$ one can efficiently compute an NFA $\overleftarrow{\mathcal{R}} = (\Sigma, Q, E, I, F)$ recognizing $\overleftarrow{R}$. Moreover, using standard constructions in automata theory, one can translate in polynomial time the LTL formula ψ into a *DFA* $\overleftarrow{\mathcal{T}}$ that recognizes $\overleftarrow{T}$, where the states are symbolically represented by vectors $\bar{q}$ of bits (one for each subformula of ψ), the transitions are symbolically represented by propositional formulas that relate the bits of the restriction state $\bar{p}$ to the bits of the target state $\bar{q}$ for any transition $(\bar{p}, a, \bar{q})$ of $\overleftarrow{\mathcal{T}}$, and, finally, the initial state and the set of final states are symbolically represented by analogous propositional formulas. The symbolic representation of the DFA $\overleftarrow{\mathcal{T}}$ has size polynomial in the size of the formula ψ. Moreover, one has to keep in mind that the DFA $\overleftarrow{\mathcal{T}}$ is not pruned, namely, it may contain states that are not reachable from the initial state or that cannot reach some final states.
We now make a couple of remarks related to algorithms on symbolically represented automata:

1) First of all, one can perform symbolic reachability analysis on the DFA $\overleftarrow{\mathcal{T}}$ in PSPACE. For instance, deciding whether $\overleftarrow{\mathcal{T}}$ contains a path between two given states $\bar{p}$ and $\bar{q}$ can be done by a non-deterministic polynomial space algorithm that guesses, step by step, the correct moves of $\overleftarrow{\mathcal{T}}$ to go from $\bar{p}$ to $\bar{q}$ (this exploits the fact that reachability between states is witnessed by paths of length at most exponential). As a matter of fact, PSPACE reachability analysis can be also done on symbolically deterministic NFA.
2) Given a SCC C of the NFA $\overleftarrow{\mathcal{R}}$ and a state $\bar{r}$ of the DFA $\overleftarrow{\mathcal{T}}$, one can decide in PSPACE whether the language $\mathscr{L}(\overleftarrow{\mathcal{R}}|C)$ in contained in the language $\mathscr{L}(\overleftarrow{\mathcal{T}}|C_{\bar{r}})$, where $CC_{\bar{r}}$ is the SCC of the state $\bar{r}$ in $\mathcal{T}$. This is done as follows. We first turn, in polynomial time, the NFA $\overleftarrow{\mathcal{R}}|C$ into an equivalent symbolic DFA $\overleftarrow{\mathcal{R}}_C$ using the usual subset construction: the states of $\overleftarrow{\mathcal{R}}_C$ are vectors of bits specifying which state of the original NFA $\overleftarrow{\mathcal{R}}|C$ is achievable; the transitions of $\overleftarrow{\mathcal{R}}_C$ are symbolically represented by suitable propositional formulas. We then compute, again in polynomial time, the symbolic representation of the complement $\overleftarrow{\mathcal{T}}^{\mathsf{C}}$ of $\overleftarrow{\mathcal{T}}$ and from there a symbolic representation of the synchronous product $\mathcal{P}$ of $\overleftarrow{\mathcal{R}}_C$ and $\overleftarrow{\mathcal{T}}^{\mathsf{C}}$ (note that the result is a symbolic DFA). To verify that $\mathscr{L}(\overleftarrow{\mathcal{R}}_C)$ is contained in $\mathscr{L}(\overleftarrow{\mathcal{T}}|C_{\bar{r}})$ it is sufficient to check that for every pair of (initial and final) states $\bar{p}, \bar{q}$ in $\overleftarrow{\mathcal{R}}_C$, there exist some states $\bar{p}'$ and $\bar{q}'$ in $\overleftarrow{\mathcal{T}}^{\mathsf{C}}$ such that (i) the three states $\bar{r}$ $\bar{p}'$, and $\bar{q}'$ are mutually reachable in $\overleftarrow{\mathcal{T}}^{\mathsf{C}}$ (this implies that $\bar{p}'$ and $\bar{q}'$ belong to the same SCC $C_{\bar{r}}$ of $\bar{r}$) and (ii) $(\bar{q}, \bar{q}')$ is reachable from $(\bar{p}, \bar{p}')$ in the DFA $\mathcal{P}$. These properties can be decided in PSPACE using symbolic reachability analysis.

We are now ready to describe the PSPACE procedure that solves the bounded repair problem between $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$. Intuitively, this is a variant of Algorithm A.1 presented in this appendix. Precisely, one first guesses universally a path $\pi = C_1 \ldots C_k$ in $\mathcal{D}ag(\overleftarrow{\mathcal{R}})$ (doable in coNP), then one guesses existentially a sequence $\bar{r}_1 \ldots \bar{r}_k$ of states of $\overleftarrow{\mathcal{T}}$ that represents a corresponding path $\pi' = C_{\bar{r}_1} \ldots C_{\bar{r}_k}$ in $\mathcal{D}ag^*(\overleftarrow{\mathcal{T}})$ (doable in NP), and finally one verifies that

i) each language $\mathscr{L}(\overleftarrow{\mathcal{R}}|C_i)$ in contained in the language $\mathscr{L}(\overleftarrow{\mathcal{T}}|C_{\bar{r}_i})$ (doable in PSPACE),
ii) each state $\bar{r}_i$ is reachable from the initial state of $\overleftarrow{\mathcal{T}}$ (doable in PSPACE),
iii) each state $\bar{r}_i$ can reach a final state of $\overleftarrow{\mathcal{T}}$ (doable again in PSPACE).

This proves that the bounded repair problem for $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$ (and hence that for $\mathcal{R}$ and ψ) is in PSPACE. ■

Theorem 10. *The bounded repair problem in the non-streaming case, where the restriction language R is represented by an LTL formula and the target language T is represented by an NFA, is in PSPACE.*

Proof: Let us fix an LTL formula ϕ defining a restriction language R and an NFA $\mathcal{T}$ recognizing a target language T. As usual, we can reverse the languages and convert ϕ (resp., $\mathcal{T}$) to a symbolic DFA $\overleftarrow{\mathcal{R}}$ (resp., to an NFA $\overleftarrow{\mathcal{T}}$). The technique for solving the bounded repair problem between $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$ is somehow similar to that used in Proposition 7. Intuitively, by exploiting symbolic reachability analysis over $\overleftarrow{\mathcal{R}}$, we move down the DAG of the SCCs of $\overleftarrow{\mathcal{R}}$, universally guessing, at each step, a state $\bar{r}$ that represents a SCC $C_{\bar{r}}$ of $\overleftarrow{\mathcal{R}}$ and maintaining the collection $F_{\bar{r}}$ of all SCCs C' of $\mathcal{T}$ such that $\mathscr{L}(\overleftarrow{\mathcal{R}}|C_{\bar{r}}) \subseteq \mathscr{L}(\overleftarrow{\mathcal{T}}|C')$. At the end of the computation (i.e., when there is no successor SCC of $\overleftarrow{\mathcal{R}}$ to move into), we anwer positively iff the reached set $F_{\bar{r}}$ is not empty.
Concerning the complexity of the above described procedure, we make the following important remarks:

1) The collection $F_{\bar{r}}$ can be stored in polynomial space (i.e., linear in the size of $\overleftarrow{\mathcal{T}}$).
2) In order for the algorithm to be correct, at each step we need to check that the guessed state $\bar{r}$ is reachable from the initial state and can be reach some final states of $\overleftarrow{\mathcal{R}}$. Moreover, the SCC $C_{\bar{r}}$ of $\bar{r}$ must be a successor in $\mathcal{D}ag(\overleftarrow{\mathcal{R}})$ of the SCC of the state that was guessed at the previous step. All these properties can be checked using symbolic reachability analysis on $\overleftarrow{\mathcal{R}}$.
3) Each step of the above described procedure requires checking a language containment $\mathscr{L}(\overleftarrow{\mathcal{R}}|C_{\bar{r}}) \subseteq \mathscr{L}(\overleftarrow{\mathcal{T}}|C')$, where $\bar{r}$ is a state of the DFA $\overleftarrow{\mathcal{R}}$, $C_{\bar{r}}$ is its SCC, and C' is a SCC of the NFA $\overleftarrow{\mathcal{T}}$. This can be done in PSPACE using a technique similar to the proof of Theorem 9, that is, by first turning $\overleftarrow{\mathcal{T}}|C'$ into a symbolic DFA $\overleftarrow{\mathcal{T}}_{C'}$, then computing a symbolic representation for the synchronous product of $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}^{\complement}_{C'}$, and finally checking that for every pair of states $\bar{p}$ and $\bar{q}$ of $\overleftarrow{\mathcal{R}}$ that belong to the same component $C_{\bar{r}}$ of $\bar{r}$, there exist some states $\bar{p}'$ and $\bar{q}'$ of $\overleftarrow{\mathcal{T}}_{C'}$ that are mutually reachable and for which $(\bar{q}, \bar{q}')$ is reachable from $(\bar{p}, \bar{p}')$ in $\mathcal{P}$. These properties can be decided in PSPACE using symbolic reachability analysis.

Since all the mentioned steps can be carried out in polynomial space, we conclude that the bounded repair problem for $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$ (and hence that for ϕ and $\mathcal{T}$) is in PSPACE. ■

Theorem 11. *The problem of determining, given k and two languages R and T recognized by DFA, whether $\mathcal{D}ist(R, T)$ is above k, is PSPACE-complete. The same holds when R and T are given as an NFA.*

Proof: We first give the upper bound, assuming that R and T are recognized by some NFA $\mathcal{R}$ and $\mathcal{T}$. Given k and word w, let us denote by $\mathsf{Impact}(w, \mathcal{R}, \mathcal{T}, k)$ the vector of the form $(R_0, T_0, \ldots T_k)$ where R_0 is the set of states that are reached by $\mathcal{R}$, starting from an initial state after reading w, and for each $9 \leq i \leq k$, T_i is the set of states of $\mathcal{T}$ achievable by editing w by i edits. Let us call a vector of the form $(R_0, T_0, \ldots T_k)$ *attainable* if it is of the form $\mathsf{Impact}(w, \mathcal{R}, \mathcal{T}, k)$ for some word w, and call it *successful* if, furthermore, some state in R_0 is accepting in $\mathcal{R}$ and all states in each set T_i are non-accepting for $\mathcal{T}$. The attainable vectors can be given an automaton structure: given an attainable vector $(R_0, T_0, \ldots T_k)$ and a letter a, we can efficiently determine the vector $(R'_0, T'_1 \ldots T'_k)$ to which we should transition – that is, the one representing $\mathsf{Impact}(wa, \mathcal{R}, \mathcal{T}, k)$ for any w such that $(R_0, T_0, \ldots T_k) = \mathsf{Impact}(w, \mathcal{R}, \mathcal{T}, k)$. R'_0 is the image of R_0 under the a-labeled transitions of $\mathcal{R}$, T'_0 is the image of T_0 under the a-labeled transitions of $\mathcal{T}$, and for each $0 \leq i < k$, T'_{i+1} is the union of the image of T_{i+1} under the a-labeled transitions of $\mathcal{T}$, T_i, the image of T_i under b-transitions of $\mathcal{T}$, for any symbol $b \neq a$, and the image of T_i under pair of consecutive transitions of $\mathcal{T}$ that consume ab, for any symbol b. We call the resulting automaton *impact-automaton* and we let $v_0 = \mathsf{Impact}(\varepsilon, \mathcal{R}, \mathcal{T}, k)$ be its initial state.
We have that $\mathcal{D}ist(\mathcal{R}, \mathcal{T})$ is above the threshold k exactly when there is a path in the impact-automaton that consists of attainable vectors starting from v_0 and leading to a successful vector. Although there are exponentially many states, each state can be represented within polynomial space and one can calculate in polynomial time the initial state v_0 and the successors of any given state. This enables reachability analysis on the impact-automaton in polynomial space and it implies the PSPACE upper bound for the reduced threshold problem.

We now discuss the PSPACE lower bound, which holds even for languages represented by DFA. It is via reduction from the problem of tiling a corridor of polynomial width (and unbounded height). An instance of the latter problem is given by a number n (i.e., the width of the corridor, represented in unary notation), a set S of available tiles, some sets $H, V \subseteq S \times S$ of vertical and horizontal constraints, and two tiles $t_\perp$ and $t_\top$ for the bottom and top rows. Hereafter, we fix an instance $I = (n, S, H, V, t_\perp, t_\top)$ of the corridor tiling problem and a threshold $k \geq 2$ (by a slight modification of the encodings described below, one could generalize the proof to the case $k \geq 1$ – note that for $k = 0$ the threshold problem becomes a containment problem between DFA, which is clearly solvable in PTIME). Moreover, we assume that the threshold k is represented in unary notation (clearly this does not make the threshold problem more difficult).

We let the restriction alphabet Σ consist of a separator symbol $\#$ and the pairs $\langle t, j\rangle$, where t is a tile from S and j is a number in $\{0, \dots, n-1\}$. A symbol $\langle t, j\rangle \in \Sigma$ can be thought of as identifying the tile t and its horizontal co-ordinate j. The restriction language R contains "k-redundant" encodings of tilings, namely, words of the form

$$\#\langle g(0,0), 0\rangle^k \dots \#\langle g(0,n-1), n-1\rangle^k \;\dots\dots\; \#\langle g(N-1,0), 0\rangle^k \# \dots \langle g(N-1,n-1), n-1\rangle^k$$

where N is a positive natural number and $g : [0, N-1] \times [0, n-1] \;\to\; S$ is a tiling function that satisfies the horizontal constraints and the constraints on the bottom and top rows (but possibly not the vertical constraints). Note that, differently from the proof of Theorem 8, the number of rows of the tiling is not fixed (the problem is still PSPACE-hard), the separator symbol $\#$ appears at the beginning, rather than at the end, of a block of symbols $\langle g(i,j), j\rangle$, and, moreover, the number of repetitions in each block is fixed to be equal to the given threshold k. We claim that the above language is recognized by a DFA $\mathcal{R}$ of size polynomial in I and in k: indeed, the automaton $\mathcal{R}$ needs to check that the input word is well-formed, which requires enforcing the horizontal constraints and the initial and final tile requirements (which clearly can be done with a fixed number of states) and counting up to k and n (which clearly can be done with a number of states linear in k and n).
We now turn to the definition of the target language. Its alphabet Δ contains all symbols of the restriction alphabet Σ plus a marking symbol $\underline{\#}$, which is used to highlight a violation of the vertical constraints. Given a word v over Δ, we call *witnessing t-block* any factor $v[x..y]$ of v that starts with the marking symbol $\underline{\#}$ and contains a k-repetition of a symbol of the form (t, j), with $0 \leq j < n$. Intuitively, a violation of a vertical constraint is certified on a word $v \in \Delta^*$ when v contains a witnessing t-block $v[x..y]$ and a witnessing t'-block $v[x'..y']$ and the infix $v[y+1..x'-1]$ delimited by these two blocks contains exactly $n-1$ occurrences of the separator symbol $\#$ (this implies that the tile t' is adjacent to t along the vertical axis). More precisely, the target language T is defined as follows:

$$T \;=\; \bigcup_{(t,t') \notin V} \Sigma^* \cdot \; \underline{\#}\langle t, j\rangle^k \cdot \big(\{\#\} \cdot (\Sigma \setminus \{\#\})^k\big)^{n-1} \cdot \; \underline{\#}\langle t', j\rangle^k \cdot \Sigma^*$$

It is easy to construct a DFA $\mathcal{T}$ of size polynomial in I and k that recognizes the language T.
We now argue that there is a tiling of a corridor $[0, N-1] \times [0, n-1]$ that satisfies the constraints in I iff there is a strategy for repairing R into T with worst-case cost at least $k+1$.
On the one hand, suppose that there is no tiling $g : [0, N-1] \times [0, n-1] \;\to\; S$ that satisfies the constraints in I. Let us consider a word $u \in R$ of the form

$$\#\langle g(0,0), 0\rangle^k \dots \#\langle g(0,n-1), n-1\rangle^k \;\dots\dots\; \#\langle g(N-1,0), 0\rangle^k \# \dots \langle g(N-1,n-1), n-1\rangle^k$$

where g is a tiling of $[0, N-1] \times [0, n-1] \;\to\; S$. From the previous assumptions, we know that g violates the constraints in I. This implies that u contains two factors $u[x..y] = \#\langle g(i,j), j\rangle^k$ and $u[x'..y'] = \#\langle g(i+1,j), j\rangle$ that are separated by exactly $n-1$ occurrences of $\#$ and such that $(g(i,j), g(i+1,j)) \notin V$. Replacing the two occurrences of $\#$ in these factors by the marking symbol $\underline{\#}$ will bring us into the target language T. This shows that there is a non-streaming strategy for repairing R into T with worst-case cost at most 2 ($\leq k$).
On the other hand, suppose that there is a tiling $g : [0, N-1] \times [0, n-1] \;\to\; S$ that satisfies the constraints in I. Let u be the corresponding word:

$$\#\langle g(0,0), 0\rangle^k \dots \#\langle g(0,n-1), n-1\rangle^k \;\dots\dots\; \#\langle g(N-1,0), 0\rangle^k \# \dots \langle g(N-1,n-1), n-1\rangle^k.$$

Clearly, u belongs to the restriction language R. We claim that no matter how we edit u by k edits, the resulting sequence will not belong to the target language T. Consider a word v produced by such an edit. If there are not two occurrences of witnessing blocks starting with $\underline{\#}$ and having $n-1$ separator symbols between them, then v cannot satisfy T. If there are two such occurrences, then the tiles encoded within these witnessing blocks must be the same as in the input word u (recall that every witnessing block has length $k+1$ and hence it cannot be edited entirely). By previous assumptions, these tiles must satisfy the vertical constraints. This shows again that v does not belong to the target language T. ■

We now turn to the results for LTL, starting with the case of word-to-formula distance:

Proposition 27. *There is a PSPACE Turing Machine that given a word w and LTL formula ϕ determines the distance of w to $L(\phi)$ in binary.*

Proof: Note that the distance of w to an NFA A is bounded by a polynomial in the size of A and w. Using this we can compute in binary an upper bound on the distance of w to an LTL ϕ, deriving it from a bound on the size of an NFA equivalent to ϕ.

Thus it suffices to solve the corresponding threshold problem: given k in binary, decide whether the distance from w to $L(\phi)$ is below k. A ϕ-type is a maximal consistent collection of subformulas of ϕ. A ϕ-type t_1 reaches a ϕ-type t_2 via word w if there are words w_1, w_2 such that $(w_1 \cdot w \cdot w_2, |w_1|) \models t_1$ and $(w_1 \cdot w \cdot w_2, |w_1| + |w|)) \models t_2$. We give a more general PSPACE algorithm for determining, given two ϕ-types t_1 and t_2, k in binary, and word w whether or not there is a word w' obtainable via at most k edits of w such that t_1 reaches t_2 via w'. The algorithm consists of a top-level alternating LOGSPACE routine that takes w t_1 and t_2 guesses an intermediate t_3 and $k' < k$, divides w into two nearly equal sized factors (rounding off) w_1 and w_2 and recursively checks that t_1 can reach t_3 via w_1 and t_3 can reach t_2 via w_2.

The alternating algorithm bottoms out at the case of $w = \epsilon$ for two types t_1 and t_2. Here it calls a subroutine performing reachability analysis in the type graph, beginning at t_1 and maintaining a counter in binary, truncating when we reach the threshold k; it is easy to see that the reachability problem can be solved in PSPACE.

The above describes an alternating LOGSPACE computation that performs a PSPACE subcomputation at the leaves; each thread of the alternating computation can be stored in PSPACE, thus the entire algorithm runs in PSPACE. ■

Theorem 12. *The problem of determining, given k and two languages R and T defined by LTL formulas, whether $\mathcal{D}ist(R, T)$ is above k, is EXPSPACE-complete.*

Proof: The upper bound follows from converting the formulas to NFA and applying Theorem 11.
The lower bound is obtained by a reduction from the problem of tiling a corridor of width 2^n. The reduction is analogous to that of Theorem 11, with the only difference that, instead of using single symbols to represent a tile and its co-ordinate, we use sequences consisting of a tile symbol and n bits for the binary encoding of the co-ordinate. Checking that the sequences of co-ordinates are well-formed (and similarly, checking that two witnessing blocks have the same co-ordinates) can be done by suitable LTL formulas similar to those used in the proof of Theorem 8. ■

C. Proofs for Section VI (Complexity results in the streaming case)

Theorem 15. *The bounded repair problem in the streaming case, where the restriction language is a DFA and the target language is in NFA is PSPACE-complete. The same result holds when the target language is restricted to be a DFA and the restriction is an NFA.*

Proof: We make use of the characterization Theorem 3 in both scenarios.
Let us first consider the case where the restriction language is given by a DFA $\mathcal{R}$ and the target language is given by an NFA $\mathcal{T}$. Let $\det(\mathcal{T})$ denotes the DFA obtained by applying the standard determinization procedure to $\mathcal{T}$. Recall that the states of $\det(\mathcal{T})$ are sets of states of $\mathcal{T}$. We can then exploit the fact that the longest collection of moves of Adam in the arena $\mathcal{A}_{\mathcal{R},\det(\mathcal{T})}$ is linear in the size of $\mathcal{D}ag(\mathcal{R})$. This implies that the length of any play is at most $|\mathcal{D}ag(\mathcal{R})|$ and hence we can simulate the reachability game on $\mathcal{A}_{\mathcal{R},\det(\mathcal{T})}$ by an alternating polynomial-time procedure. Precisely, we can keep track of the configuration of the reachability game by maintaining the current SCC C of $\mathcal{R}$ and (a symbolic representation of) the current SCC C' of $\det(\mathcal{T})$ (note that a SCC of $\det(\mathcal{T})$ can be represented by a single state of $\det(\mathcal{T})$ or, equivalently, by a set of states of $\mathcal{T}$). At each round of the reachability game, we need to check a language containment $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\det(\mathcal{T})|C')$: this can be done using the characterization given in Lemma 25 and a PSPACE subroutine based on symbolic reachability analysis. What we have described is an alternating polynomial-time procedure that simulates the reachability game over $\mathcal{A}_{\mathcal{R},\det(\mathcal{T})}$ using a PSPACE subroutine for language containment. Overall, the resulting complexity is in PSPACE.

We turn to the case where the restriction language is given by an NFA $\mathcal{T}$ and the target language is given by a DFA $\mathcal{T}$. As in the previous proof, the general idea is to simulate the reachability game over the arena $\mathcal{A}_{\det(\mathcal{R}),\mathcal{T}}$ that results from the main characterization result. However, we cannot obtain a polynomial bound to the length of the plays since the DAG of $\mathcal{R}$ has potentially exponential height. The crucial observation here is that it is possible to modify the definition of the arena $\mathcal{A}_{\det(\mathcal{R}),\mathcal{T}}$ (and thus the resulting reachability game) by allowing Adam to move down the DAG of $\det(\mathcal{R})$ with shortcuts, namely, by allowing Adam to move from any SCC to some descendant of it (rather than simply a successor of it). On the one hand, allowing this freedom in the new reachability game clearly makes it easier for Adam to win. We argue that, if he wins, he can do it within a polynomial number of moves. Indeed, suppose that Adam wins in the original arena, then he can also win in the modified arena and, moreover, by properly choosing shortcut moves, he can push Eve towards a sink node in at most n rounds, where n is the height of the DAG of SCCs of $\mathcal{T}$. On the other hand, if Adam wins in the modified arena, then he can also win in the original arena via longer plays. The above arguments show that the two versions of the reachability games are equivalent and, furthermore, one can bound the length of the plays to a polynomial in the size of the DFA $\mathcal{T}$. Therefore, the bounded streaming repair problem for $\mathcal{R}$ and $\mathcal{T}$ can be solved by an alternating polynomial time procedure similar to the one described above. This shows that the problem is in PSPACE. ■

Theorem 16. *The bounded repair problem in the streaming case, where the restriction and target languages are represented by LTL formulas, is in 2EXPTIME and is EXPSPACE-hard.*

Proof: As already mentioned, the 2EXPTIME complexity upper bound follows from standard constructions in automata theory one can translate any two given LTL formulas ϕ and ψ into NFA $\mathcal{R}$ and $\mathcal{T}$ recognizing the languages defined by ϕ and ψ, respectively. The NFA $\mathcal{R}$ and $\mathcal{T}$ have size exponential in $|\phi|$ and $|\psi|$. Moreover, by using the classical determinization procedure, one can turn the NFA $\mathcal{R}$ and $\mathcal{T}$ to equivalent DFA $\mathcal{R}'$ and $\mathcal{T}'$, which have size at most doubly exponential in $|\phi|$ and $|\psi|$. Corollary 13 then implies that the problem of deciding whether there is a streaming repair processor of the language defined by ϕ into the language defined by ψ with worst-case finite cost is in 2EXPTIME.

We now turn to the proof of the lower bound; we reduce the problem of deciding the winner of a tiling game on a grid of exponential size to the bounded repair problem in the streaming case. We first describe the tiling game. As in the proof of Theorem 8, an instance of the tiling game is a tuple $I = (n, S, H, V, t_\perp, t_\top)$, where S is a finite set of tiles, $H, V \subseteq S \times S$ are the relations for the horizontal and vertical constraints between tiles, 2^n is the length of the edge of the square grid $\mathcal{G}_n = [0, 2^n - 1] \times [0, 2^n - 1]$ to be tiled (as usual, n is represented in unary notation), and $t_\perp$ and $t_\top$ are the tiles that must appear at the bottom and top rows of the grid. Given a number $0 \leq k \leq 2^n$, we define a *tiling of height* k (for the instance I) to be any function g that maps the pairs (i, j) in $\mathcal{G}_n$, with $0 \leq i < k$ and $0 \leq j < 2^n$, to the tiles in S and that, furthermore, satisfies $g(0, j) = t_\perp$ if $k > 0$ and $g(2^n - 1, j) = t_\top$ if $k = 2^n - 1$, for all $0 \leq j < 2^n$. We say that a tiling g of height k is *correct* if it satisfies the constraints of I, namely, if for every pair of indices $0 \leq i < k$ and $0 \leq j < 2^n$, with $j > 0$ (resp., $i > 0$), we have $\big(g(i, j-1), g(i, j)\big) \in H$ (resp., $\big(g(i-1, j), g(i, j)\big) \in V$). The tiling game for the instance I is played between two players, Eve (who corresponds to the target language) and Adam (who corresponds to the restriction language), as follows. A configuration of the tiling game at round k, with $0 \leq k \leq 2^n$, is a correct tiling g_k of height k (hence the empty tiling of height 0 is the initial configuration of the game). Adam moves at even rounds (hence Adam moves first), while Eves moves at odd rounds. Given a correct tiling g_k at round k, the move of the corresponding player consists of extending g_k to a correct tiling g_{k+1} of height $k+1$. The player who cannot move loses (hence Adam loses if the last round $k = 2^n$ is reached). We know from [8] that the problem of deciding the winner of a tiling game is AEXPTIME-hard (hence EXPSPACE-hard). Below, we fix a tiling instance $I = (n, S, H, V, t_\bot, t_\top)$ and we construct suitable LTL formulas ϕ and ψ that have size polynomial in $|I|$ and that define two regular languages R and T such that R can be repaired into T by a streaming strategy with uniformly bounded cost iff Eve wins the tiling game associated with the instance I. This would imply that the bounded repair problem for LTL formulas in the streaming case is EXPSPACE-hard.

The basic idea of the reduction is similar to the proof of Theorem 8. Intuitively, the player Adam (resp., Eve) corresponds to a processor that produces a (redundant) encoding of a partial tiling for the even (resp., odd) rows in $\mathcal{G}_n$. Moreover, in order to check that the resulting tilings are correct, the two processors can emit repetitions of a distinguished block that make it easy to certify the violation of some horizontal or vertical constraints of I. The alphabets Σ and Δ for the restriction and target languages consist of the symbols $0, 1$ (which are used to encode the tile coordinates), the symbol $\#$ (which is used to separate the various blocks), an underlined copy $\underline{\#}$ of $\#$ (which is used to decorate the block that witnesses a violation), the symbol $\square$ (which is used to separate the various rows).

The restriction language R, which corresponds to Adam's objective, contains the encodings of the tilings of $\mathcal{G}_n$ along the even rows, up to some height, extended with repetitions of a distinguished underlined block witnessing a possible violation. We define R to be the union, over all even numbers $0 \le k < 2^n - 1$, all tilings g of height $k+2$, and all indices $0 \le j' < 2^n$, of the languages

$$L^{\text{odd}}_{k,g,j'} \;=\; \{u_{0,0}\}^+ \cdot \ldots \cdot \{u_{0,N-1}\}^+ \cdot \{\square\} \cdot \{u_{2,0}\}^+ \cdot \ldots \cdot \{u_{2,N-1}\}^+ \cdot \{\square\} \cdot \ldots \ldots \cdot \{u_{k,0}\}^+ \cdot \ldots \cdot \{u_{k,N-1}\}^+ \cdot \{\square\} \cdot \{\underline{u}_{k+1,j'}\}^+$$

where, as in the proof of Theorem 8, each word $u_{i,j}$ is defined as $\langle i,j\rangle g(i,j)\#$ (similarly, $\underline{u}_{k+1,j'} = \langle k+1, j'\rangle g(k+1,j')\underline{\#}$) and each word $\langle i,j\rangle$ is the juxtaposition of the binary encodings of the coordinates i and j. The target language T, which corresponds to Eve's objective, contains the encodings of the tilings of $\mathcal{G}_n$ along all rows up to some height, possibly extended with some repetitions of a distinguished underlined block. T is the union of the languages

$$L_{k,g,j'} \;=\; \{u_{0,0}\}^+ \cdot \ldots \cdot \{u_{0,N-1}\}^+ \cdot \{\square\} \cdot \{u_{1,0}\}^+ \cdot \ldots \cdot \{u_{1,N-1}\}^+ \cdot \{\square\} \cdot \ldots \ldots \cdot \{u_{k,0}\}^+ \cdot \ldots \cdot \{u_{k,N-1}\}^+ \cdot \{\square\} \cdot \{\underline{u}_{k,j'}\}^* \cdot \Delta^*$$

for all numbers $0 \le k < 2^n$, all tilings g of height $k+1$, and all indices $0 \le j' < 2^n$ such that

i) if k is even, then $\big(g(k, j'-1), g(k,j')\big) \notin H$ or $\big(g(k-1, j'), g(k,j')\big) \notin V$ (namely, Eve witnessed a violation of the constraints in the previous move of Adam).

ii) if k is odd, then $\big(g(k, j'-1), g(k,j')\big) \in H$ and $\big(g(k-1, j'), g(k,j')\big) \in V$ (namely, Adam did not witness any violation of the constraints in previous move of Eve),

Using constructions similar to those in the proof of Theorem 8, one can define the restriction and target languages R and T by means of suitable LTL formulas ϕ and ψ of size polynomial in the instance I. We omit the formal definitions of these formulas and we focus instead on the reduction of the tiling game.

Suppose that Eve has a winning strategy in the tiling game. This strategy can be described as a function s that maps a correct tiling g of odd height $1 \le k < 2^n$ to a correct tiling g' of height $k+1$ that extends g. Using this function s, we can construct a transducer $\mathcal{S}$ that realizes a streaming repair strategy of R into T with uniformly bounded cost. Here we omit the formal definition of such a transducer, which is tedious, and we only describe its behaviour informally. The computation of the transducer is divided into 2^n phases, which correspond to the 2^n rounds of the tiling game; the behaviour of the transducer is defined according to the parity of the phase. At the beginning of an *even* phase k (including the initial phase $k = 0$), the state of the transducer $\mathcal{S}$ represents the current configuration of the tiling game at round k, which consists of a *correct* tiling g_k of height k. During this phase, the transducer $\mathcal{S}$ consumes (without modifying) a portion of the input word until it identifies one of the following two sub-sequences in it:

1) $\langle k, 0\rangle g_{k+1}(k,0)\# \cdot \ldots \cdot \langle k, 2^n - 1\rangle g_{k+1}(k, 2^n - 1)\# \cdot \square$, for some tiling g_{k+1} of height $k+1$ that extends g_k (note that the tiling g_{k+1} may not be correct),

2) $\langle k-1, j'\rangle g_k(k-1, j')\underline{\#}$, for some index $0 \le j' < 2^n$.

In the first case, the transducer moves to the next (odd) phase $k+1$ by storing the new tiling g_{k+1}. In the second case, since the tiling g_k is correct, we know that $\big(g_k(k-1, j'-1), g_k(k-1, j')\big) \in H$ and $\big(g_k(k-2, j'), g_k(k-1, j')\big) \in V$. In this case, the transducer can directly move to a sink state that reproduces any input symbol later on (observe that this kind of behaviour guarantees that the transducer eventually outputs a word in the target language T). During an *odd* phase k, the transducer remembers the current tiling g_k of height k, which is guaranteed to be correct up to height $k-1$. The transducer here distinguishes two cases, depending on whether the tiling g_k is entirely correct or not. If g_k is correct, then the transducer emits the encoding $\langle k, 0\rangle g_{k+1}(k,0)\# \cdot \ldots \cdot \langle k, 2^n - 1\rangle g_{k+1}(k, 2^n - 1)\# \cdot \square$ of the tiling g_{k+1} along the row k that is specified by Eve's strategy (i.e., $g_{k+1} = s(g_k)$) and it accordingly moves to the next (even) phase $k+1$. Otherwise, if g_k is not correct, then we know that there is an index $0 \le j' < 2^n$ such that $\big(g_k(k-1, j'-1), g_k(k-1, j')\big) \notin H$ or $\big(g_k(k-2, j'), g_k(k-1, j')\big) \notin V$. In this case, the transducer emits the word $\langle k-1, j'\rangle g_k(k-1, j')\underline{\#}$ and moves to a sink

state that reproduces any input symbol later on (this guarantees that the output word belongs to the target language T). From the above arguments it is clear that the transducer $\mathcal{S}$ realizes a streaming strategy for repairing the restriction language R into the target language T with uniformly bounded cost (note that, even if the worst-case cost depends on the instance I, it is uniformly bounded once we fix I and the corresponding languages R and T).
We now prove the converse implication. We assume that Adam has a winning strategy in the tiling game and we describe this strategy by a function s that maps a correct tiling g of even height $0 \le k < 2^n - 1$ to a correct tiling g' of height $k+1$ that extends g. Moreover, by way of contradiction, we assume that there is a transducer $\mathcal{S}$ that realizes a streaming repair strategy of R into T with cost uniformly bounded by some natural number N. For the sake of brevity, we say that a word u encodes a tiling g along the row i if u contains a sub-sequence of the form $\langle i,0\rangle g(i,0)\# \cdot \ldots \cdot \langle i, 2^n-1\rangle g(i,2^n-1)\# \cdot \Box$.
We recursively construct some tilings $g_0, g_1, g_2, ..., g_k$ and some words $v_1, v_3, ..., v_{k+1}$ such that

i) for all even indices $2 \le i \le k$, g_i is a tiling of height i that extends g_{i-1},

ii) for all odd indices $1 \le i < k$, g_i is a *correct* tiling of height i that extends g_{i-1},

iii) for all odd indices $1 \le i < k$, v_i encodes the tiling g_i along the row $i-1$,

iv) for all odd indices $1 \le i \le k+1$, the word $v_1 \cdot v_3 \cdot \ldots \cdot v_i$ belongs to the restriction language R,

v) $\mathcal{S}(v_1 \cdot v_3 \cdot \ldots \cdot v_{k+1})$ does not belong to the target language T.

As for the base case, we simply let g_0 be the tiling of height 0. As for the inductive case, we assume that there exist some tilings $g_0, g_1, g_2, ..., g_i$ and some words $v_1, v_3, ..., v_{i-1}$, for some even index $i \ge 0$, that satisfy the properties i)–iv) above and we define the word v_{i+1} and, possibly, the tilings g_{i+1} and g_{i+2} as follows. First, we distinguish between two cases, depending on whether g_i is correct or not.
If g_i is not correct, then we necessarily have $i \ge 2$ and hence g_i extends the tiling g_{i-1}, which is correct. This shows that there is an index $0 \le j' < 2^n$ such that $\big(g_i(i-1,j'-1), g_i(i-1,j')\big) \notin H$ or $\big(g_i(i-2,j'), g_i(i-1,j')\big) \notin V$. In this case, we simply let $k = i+2$ and v_{i+1} be the underlined block $\langle i,j'\rangle g_{i+1}(i,j')\underline{\#}$. This concludes the inductive definition and it guarantees that the property v) above is also verified.
In the second case, we assume that the tiling g_i is correct. We then use Adam's strategy to define a the correct tiling $g_{i+1} = s(g_i)$ of height $i+1$ that extends g_i. Accordingly, we let v_{i+1} be a redundant encoding of the tiling g_{i+1} along the row i:

$$v_{i+1} \;=\; \big(\langle i,0\rangle g_{i+1}(i,0)\#\big)^{N+1} \cdot \ldots \cdot \big(\langle i,2^n-1\rangle g_{i+1}(i,2^n-1)\#\big)^{N+1}.$$

Since v_{i+1} contains more than N occurrences of each block $\langle i,j\rangle g_{i+1}(i,j)\#$, the output of the transducer $\mathcal{S}$ on v_{i+1} must contain a sub-sequence that encodes the tiling g_{i+1} along the row i. By analyzing the definition of the target language T, we see that the transducer has, basically, only two options for repairing the word v_{i+1}. The first option is to emit an underlined block of the form $\langle i,j'\rangle g_{i+1}(i,j')\underline{\#}$. In this case, however, we have $\big(g_{i+1}(i,j'-1), g_{i+1}(i,j')\big) \in H$ and $\big(g_{i+1}(i-1,j'), g_{i+1}(i,j')\big) \in V$ (recall that the tiling g_{i+1} is correct) and hence the repaired word $\mathcal{S}(v_1 \cdot v_3 \cdot \ldots \cdot v_{i+1})$ does not belong to the target language T. The second option is to emit an encoding of a tiling g_{i+2} of height $i+2$ that extends g_{i+1}, provided that $i+2 \le 2^n$. This latter case concludes the inductive definition.
The above arguments show that there is a word $v_1 \cdot v_3 \cdot \ldots \cdot v_{k+1}$ in the restriction language that is not repaired by $\mathcal{S}$ into a word in the target language. This is against the hypothesis of $\mathcal{S}$ realizing a streaming repair strategy of R into T with uniformly bounded cost. ■

Theorem 17. *The problem of determining, given k and two languages R and T recognized by DFA, whether one can repair R into T in the streaming case with aggregate cost at most k, is in PTIME.*

Proof: Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', q'_0, F')$ be two DFA, let R and T be the recognized languages, and let k be the repair threshold. As a preliminary remark, we observe that, without loss of generality, we can assume that k is represented in unary: indeed, we know from Theorem 3 that either there is a streaming strategy for repairing R into T with aggregate cost uniformly bounded by $(1 + |\mathcal{D}ag(\mathcal{R})|) \cdot |\mathcal{T}|$ (i.e., a polynomial in the size of $\mathcal{R}$ and $\mathcal{T}$), or all streaming repair strategies of R into T have unbounded aggregate cost.
We define a reachability game over an arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ that characterizes the threshold problem for $\mathcal{R}$ and $\mathcal{T}$ in the streaming case. The nodes of the arena are either the pairs $(q, q', c) \in Q \times Q' \times [0,k]$ or the pairs $(q, q', c, a) \in Q \times Q' \times [0,k] \times \Sigma$. The former nodes are owned by Adam (i.e., the player entitled to emit a word in the restriction language) and the latter nodes are owned by Eve (i.e., the player entitled to repair the given word into the target language). The arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ has an edge $(p, p', c) \to (q, p', c, a)$ if $\delta(p,a) = q$, and an edge $(q, p', c, a) \to (q, q', c')$ if q' is reachable from p' in $\mathcal{T}$ and $c' = c + \min\big\{\mathcal{D}ist(a,v) \,:\, \delta(p',v) = q'\big\}$ (provided that $c' \le k$). Adam plays first, starting from the node $(q_0, q'_0, 0)$. The player who cannot move loses. Finally, Eve wins in all infinite plays (which are feasible in this type of game).
Now, we show that Eve has a winning strategy in $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ iff there is a streaming repair strategy of R into T with aggregate cost at most k. Easily, assume that Eve has a winning strategy in $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. This implies that there exist a winning positional

strategy for Eve. It is easy to construct, using the positional strategy of Eve, a deterministic transducer that repairs $\mathcal{R}$ into $\mathcal{T}$ with aggregate cost less than or equal to k. For the other direction, suppose that there exists a deterministic transducer $\mathcal{S} = (\Sigma, \Delta, Q'', \delta'', q_0'', \Omega)$ that repairs every word from $\mathcal{R}$ into $\mathcal{T}$ with aggregate cost less than or equal to k. It is straightforward to define a winning strategy for Eve using the deterministic transducer. Indeed, we only need to maintain the current state $s \in \mathcal{S}$ of the transducer and move from each node (q, p', c, a) to a successor node (q, q', c') such that $q' = \delta(p', v)$, $c' = c + \mathcal{D}ist(a, v)$, and $\delta''(s, a) = (v, s')$. ■