Bounded repairability of word languages

Michael Benedikt, Gabriele Puppis, Cristian Riveros

Bounded Repairability of Word Languages

Michael Benedikt[a,*], Gabriele Puppis[b], Cristian Riveros[a]

[a]Department of Computer Science - University of Oxford
[b]CNRS / LaBRI - University of Bordeaux

Abstract

What do you do if a computational object (e.g. program trace) fails a specification? An obvious approach is to perform a *repair*: modify the object minimally to get something that satisfies the constraints. This approach has been investigated in the database community, for integrity constraints, and in the AI community for propositional logics. Here we study how difficult it is to repair a document in the form of a string. Specifically, we consider number of edits that must be applied to an input string in order to satisfy a given target language. This number may be unbounded; our main contribution is to isolate the complexity of the *bounded repair problem* based on a characterization of the regular languages that admit bounded repairr. We consider the settings where the repair strategy is unconstrained and when the editing must be produced in a streaming way, i.e. by a letter-to-letter transducer.

Keywords: Bounded repair, edit distance, regular languages.

1. 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 study repairs of words. In this case, 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 another. Edit distance is lifted in a natural way to give a measure of the distance of a string $\mathfrak{u}$ to a language (collection

*Corresponding author

of strings) L. It is well known [3] that the standard dynamic programming approach to edit distance extends to give an efficient algorithm for calculating the minimum distance of $\mathfrak{u}$ to any string in L, when L is a regular language given as a non-deterministic finite state automaton.

In this work we take the next step and consider a notion of "distance" between languages: given two languages R and T (specified in terms of automata), we aim to calculate how difficult it is to transform a string satisfying R into a string satisfying T. The notion is motivated by considering R to be a *restriction* – a constraint that the input is guaranteed to satisfy – while T is a *target* – a constraint that we want to enforce. We consider the worst-case over any string $\mathfrak{u} \in \mathsf{R}$ of the number of edit operations needed to move $\mathfrak{u}$ into T, that is, we look at the worst-case number of edits 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*, namely, to determine whether the supremum above is finite. In order to solve this effectively, we need to restrict the languages R and T. We consider this problem when the restriction and target languages are presented by both deterministic and non-deterministic finite state automata. In 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 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 and for any of type of representation of the languages considered in the non-streaming setting.

The above deals with the problem of determining whether the distance between two specifications 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 procedure that given a word $\mathfrak{u}$ runs a dynamic programming algorithm to compute the edit distance to the target language (e.g. the algorithm from [3]). However, 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 settings, and also give procedures for computing the optimal processor in the streaming setting.

The streaming and non-streaming repair problems have very different flavours: the former are closely related to games played on the components of two 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:

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

Organization. Section 2 gives the preliminaries. Section 3 defines the basic problems. Section 4 gives the characterizations of bounded repairability that we will use throughout the remainder. Section 5 analyses the complexity of the bounded repair problem in the non-streaming and streaming settings. Section 6 shows some connections with games and distance automata. Section 7 gives the conclusions.

Acknowledgments. We thank the anonymous referees of JCSS for many helpful comments. The authors were supported by the Engineering and Physical Sciences Research Council (UK) grant EP/G004021/1, *Enforcement of Constraints on XML Streams*.

2. Preliminaries

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

Automata. A *non-deterministic finite state automaton* (shortly, *NFA*) is a tuple 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 [4]. $\mathscr{L}(\mathcal{A})$ is the language recognized by $\mathcal{A}$. If $\mathcal{A}$ is a *deterministic finite state automaton* (*DFA*), which never has more than one applicable transition, 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 $\delta(q, a\, u) = \delta(q', u)$ iff $(q, a, q') \in E$. We naturally extend the transition function δ of a DFA from letters to strings – that is, we let $\delta(q, \varepsilon) = q$ and $\delta(q, u \cdot a) = \delta(\delta(q, u), a)$ for all $u \in \Sigma^*$.

For technical reasons, it is convenient to assume that an automaton is *trimmed*, namely, all its states are reachable from some initial states (i.e., they are accessible) and they can reach some final states (i.e., they are co-accessible). It is worth noticing that, since the decision problems we are going to deal with are at least NLOGSPACE-hard and since states of automata that are not accessible or not co-accessible can be pruned using a simple non-deterministic logarithmic-space reachability analysis, this assumption will have no impact on our complexity results.

Since automata can be viewed as directed labelled graphs, we inherit standard definitions and constructions in graph theory. In particular, given an automaton $\mathcal{A} = (\Sigma, Q, E, I, F)$ and a state $q \in Q$, we denote by $\mathcal{C}(q)$ the *strongly connected component* (SCC, for short) of $\mathcal{A}$ that contains all states mutually reachable with q. Given a set C of states of $\mathcal{A}$ (e.g., a SCC), we denote by $\mathcal{A}|C$ the NFA that recognizes the set of all words that can be consumed entirely inside the SCC C: this automaton is 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 state without transitions to itself, then the language $\mathscr{L}(\mathcal{A}|C)$ recognized by the subautomaton $\mathcal{A}|C$ is equal to $\{\epsilon\}$. Finally, we denote by $\mathsf{dag}(\mathcal{A})$ the directed acyclic graph of the SCCs of $\mathcal{A}$ and by $\mathsf{dag}^*(\mathcal{A})$ the graph obtained from the symmetric and transitive closure of the edges of $\mathsf{dag}(\mathcal{A})$.

Transducers. A (subsequential) *transducer* is a tuple $\mathcal{S} = (\Sigma, \Delta, Q, \delta, q_0, \Omega)$, where Σ is a finite input alphabet, Δ is a finite output alphabet, Q is a finite set of states, δ is a partial transition function from $Q \times \Sigma$ to $\Delta^* \times Q$, q_0 is an initial state, and Ω is a partial function from Q to Δ^*. For every input word $u = a_1 \dots a_n \in \Sigma^*$, there is at most one run of $\mathcal{S}$ on u of the form

$$q_0 \xrightarrow{a_1/v_1} q_1 \xrightarrow{a_2/v_2} \dots \xrightarrow{a_n/v_n} q_n \xrightarrow{\varepsilon/v_{n+1}}$$

where $\delta(q_i, a_i) = (v_i, q_{i+1})$ for all $0 \le i < n$ and $\Omega(q_n) = v_{n+1}$ [5]. In such a case, we define the *output* of $\mathcal{S}$ on u to be the word $\mathcal{S}(u) = v_1 v_2 \dots v_n v_{n+1}$. We observe that the transducer may append a string v_{n+1} at the end of the computation: this is precisely the notion of subsequentiality introduced in [5].

Transducers as above produce a portion of the output 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 defined as above, but 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 $1 \le i \le |u|$ 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 a 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 $q_0, \dots, q_n$ satisfying $\delta(q_i, \vec{u}_i) = (v_i, q_{i+1})$, for all $1 \le i \le n$, and $\Omega(q_n) = v_{n+1}$. Clearly, a 0-lookahead transducer is simply a standard (subsequential) transducer.

3. The bounded repair problem

Given two words $u \in \Sigma^*$ and $v \in \Delta^*$, we denote by

$$\mathsf{dist}(u, v)$$

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 [3].

We are interested in quantifying how difficult it is to edit a word in one language to obtain a word in another language. That is, we have finite alphabets Σ and Δ and regular languages $R \subseteq \Sigma^*$ and $T \subseteq \Delta^*$, called the *restriction* and *target* languages, respectively. We would like to edit any string that is known to belong to the restriction language R into a string in the target language T. How do we measure the cost of edits needed to get from R to T? One method is to look at the largest number of edit operations that are needed to get into T from strings in R, that is, the supremum over all $u \in R$ of the minimum over all $v \in T$ of $\mathsf{dist}(u, v)$:

$$\mathsf{cost}(R, T) \ =^{\text{def}} \ \sup_{u \in R} \min_{v \in T} \mathsf{dist}(u, v).$$

Intuitively, having a finite uniform bound to the edit distance of words in R from T, means that the language R is "quite close to being a subset" of T – the gap between strings in R and strings in T is small.

A *repair strategy* for two languages R and T is any function from R to T. For a repair strategy f and a word $u \in R$, we define the *cost of* f *on* u, denoted $\mathsf{cost}(u, f)$, as the edit distance between u and $f(u)$. Accordingly, we define the *worst-case cost of* f as the *supremum* of the cost of f over all words in R (if the cost of f on $u \in R$ is unbounded, then write $\mathsf{cost}(R, f) = \infty$). Note that the value $\mathsf{cost}(R, T)$ introduced in the previous paragraph can be equally described as the minimum over all repair strategies f for R and T of the worst-case cost of f: indeed, the best repair strategy for R and T is just to output on any $u \in R$ the word in T that is closest to u with respect to the edit distance.

Example 1. *Consider the regular languages* $R = a^*\ b^*$ *and* $T = a^*\ c\ b^*$. *Clearly, any string in* R *can be converted to a string in* T *with at most* 1 *edit operation, namely,* $\mathsf{cost}(R, T) = 1$*; a repair strategy that achieves this cost maps any word* $a^n\ b^m \in R$ *to the word* $a^n\ c\ b^m \in T$.

The *bounded repair problem* is to decide, given two regular languages R and T, whether $\mathsf{cost}(R, T)$ is finite or not, namely, whether all words in R can be edited into T with at most some cost bound that does not depend on the input word. A variant of the bounded repair problem, called *threshold repair problem*, consists of deciding whether $\mathsf{cost}(R, T) \leq \theta$ for two given regular languages R and T and for a given number θ. This problem is the decision version of the problem of computing the exact value of $\mathsf{cost}(R, T)$.

The regular languages R and T are naturally presented by means of automata. In the following, we will study the complexity of the bounded repair problem for languages specified by (i) deterministic finite state automata (DFA) and (ii) non-deterministic finite state automata (NFA).

Streaming vs non-streaming. The notion of "how much does it cost to edit a word in R to a word in T" assumes that a repair strategy could be any mapping from R to T (in principle, such a mapping could even fail to be computable). However, we know from [3] that there is a dynamic programming algorithm that, given a word u and a regular target language T specified by means of a finite state automaton $\mathcal{T}$, computes in time $\mathcal{O}(|u|\cdot|\mathcal{T}|)$ an optimal edit sequence that transforms u into some word 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 we can repair R into T with a one-pass algorithm, that is, using a transducer. Recall that a transducer defines a word-to-word partial function; if this function happens to produce a word in T for every input $u \in \mathsf{R}$, then we say that it is a *streaming repair strategy* for R and T. Similarly, we can consider k-lookahead transducers, with $k \in \mathbb{N}$. This type of transducer outputs words on the basis of its current state and an input $(k+1)$-character window that represents a substring of u of the form $u[i]\dots u[i+k]$, where $u[i]$ is either the i-th symbol of w, if $i \le |u|$, or a dummy symbol $\perp$, if $i > |u|$. Accordingly, we talk about a k-*lookahead streaming repair strategy* for R and T.

Given a (k-lookahead) streaming edit strategy $\mathcal{S}$ for R and T and given a word $u \in \mathsf{R}$, we can define the *cost of* $\mathcal{S}$ *on* u in two ways:

1. 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, denoted $\mathsf{cost}^{\mathsf{aggr}}_{\mathcal{S}}(u)$, to be the length of the final output v_{n+1} plus the sum, over all indices $1 \le i \le n$, of $\mathsf{dist}(a_i, v_i)$, where $\mathsf{dist}(a_i, v_i)$ is 1 if v_i is empty, $|v_i| - 1$ if a_i occurs in v_i, and $|v_i|$ otherwise;
2. considering the transducer $\mathcal{S}$ as a repair strategy, we define the *edit cost of* $\mathcal{S}$ *on* u, denoted $\mathsf{cost}^{\mathsf{edit}}_{\mathcal{S}}(u)$, to be simply the edit distance between u and the output $\mathcal{S}(u)$.

The first notion of cost considers the distortions performed in producing the input from the output – it is equivalent to considering the transducer as producing edit sequences rather than strings and counting the number of edits produced. The second notion of cost is global and it considers only the output and not its production (clearly, the edit cost never exceeds the aggregate cost). These two models of cost can be very different in general. As an example, consider a transducer $\mathcal{S}$ on the input alphabet $\Sigma = \{a, b\}$ that converts a's to b's, and b's to a'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.

Similarly, we define the *worst-case aggregate/edit cost* of the k-lookahead

streaming repair strategy $\mathcal{S}$ as follows:

$$\mathsf{cost}^{\lambda}_{\mathcal{S}}(\mathsf{R}) \ =^{\text{def}} \ \sup_{\mathfrak{u}\in\mathsf{R}} \ \mathsf{cost}^{\lambda}_{\mathcal{S}}(\mathfrak{u}) \qquad\qquad \text{for } \lambda \in \{\mathsf{aggr}, \mathsf{edit}\}.$$

The k-*lookahead streaming aggregate/edit cost* $\mathsf{cost}^{\lambda}_{\mathsf{k-lookahead}}(\mathsf{R},\mathsf{T})$ for two languages R and T and for $\lambda \in \{\mathsf{aggr}, \mathsf{edit}\}$ is then defined as the *minimum* of $\mathsf{cost}^{\lambda}_{\mathcal{S}}(\mathsf{R})$ taken over all k-lookahead streaming strategies $\mathcal{S}$ for R and T.

The following example illustrates the difference between bounded repairability in the streaming and non-streaming settings:

Example 2. *Consider the languages* $\mathsf{R} = (\mathfrak{a}+\mathfrak{b})\ \mathfrak{c}^*\ (\mathfrak{a}^*+\mathfrak{b}^*)$ *and* $\mathsf{T} = \mathfrak{a}\ \mathfrak{c}^*\ \mathfrak{a}^* + \mathfrak{b}\ \mathfrak{c}^*\ \mathfrak{b}^*$. *In the non-streaming setting, one can get from* R *to* T *by only editing the initial letter, and thus* $\mathsf{cost}(\mathsf{R},\mathsf{T}) = 1$. *In contrast, a* k-*lookahead streaming edit strategy must decide whether to leave or change the initial letter, and thus it could be forced to repair an unbounded sequence of* $\mathfrak{a}$*'s or* $\mathfrak{b}$*'s after the sequence of* $\mathfrak{c}$*'s. In this case we have* $\mathsf{cost}^{\mathsf{aggr}}_{\mathsf{k-lookahead}}(\mathsf{R},\mathsf{T}) = \mathsf{cost}^{\mathsf{edit}}_{\mathsf{k-lookahead}}(\mathsf{R},\mathsf{T}) = \infty$.

The *bounded repair problem in the* k-*lookahead streaming setting* consists of deciding whether $\mathsf{cost}^{\lambda}_{\mathsf{k-lookahead}}(\mathsf{R},\mathsf{T}) < \infty$ for a given pair of regular languages R and T, a given $\lambda \in \{\mathsf{aggr}, \mathsf{edit}\}$, and a given $\mathsf{k} \in \mathbb{N}$. 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 setting*. Even though the two models of aggregate and edit cost for streaming repair strategies can be very different, it will turn out that for the bounded repair problem it does not matter which cost model we choose (see Corollary 4.6).

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

4. Characterizations of bounded repairability

We fix a restriction language R and a target language T and we assume, for the moment, that these languages are recognized by two NFA $\mathcal{R}$ and $\mathcal{T}$, respectively. Recall that, given a SCC C of $\mathcal{R}$, we denote by $\mathcal{R}|\mathsf{C}$ the sub-automaton of $\mathcal{R}$ that is obtained by declaring the states in C to be both initial and final and by discarding all other states. Moreover, $\mathsf{dag}(\mathcal{R})$ denotes the directed acyclic graph of the SCCs of $\mathcal{R}$ and $\mathsf{dag}^*(\mathcal{T})$ denotes the symmetric and transitive closure of $\mathsf{dag}(\mathcal{T})$. Finally, recall that both $\mathcal{R}$ and $\mathcal{T}$ are trimmed, namely, unreachable and sink states are removed from $\mathcal{R}$ and $\mathcal{T}$.

4.1. Non-streaming setting

In order to characterize the positive instances of the bounded repair problem, we need to consider the relationships between the paths in $\mathsf{dag}(\mathcal{R})$ and the

paths in $\mathsf{dag}^*(\mathcal{T})$. We say that a path $\pi = C_1 \dots C_k$ of SCCs in $\mathsf{dag}(\mathcal{R})$ is *covered* by a path $\tau = D_1 \dots D_h$ of SCCs in $\mathsf{dag}^*(\mathcal{T})$ if we have $k = h$ and $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|D_i)$ for all indices $1 \le i \le k$, namely, if the language recognized by the i-th component along π is contained in the language recognized by the i-th component along τ.

The following characterization reduces the bounded repair problem in the non-streaming setting to a path coverability problem between finite directed acyclic graphs.

Theorem 4.1. *Given two NFA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

1. $\mathsf{cost}(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})) < \infty$
   (i.e., there is a strategy that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with a uniformly bounded number of edits),
2. *every path in $\mathsf{dag}(\mathcal{R})$ is covered by some path in $\mathsf{dag}^*(\mathcal{T})$,*
3. $\mathsf{cost}(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})) \le (1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|$.

Proof of Theorem 4.1. Let $\mathcal{R} = (\Sigma, Q, E, I, F)$ and $\mathcal{T} = (\Delta, Q', E', I', F')$ be two NFA. 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).

We briefly outline the main ideas underlying the proof of the implication from 2 to 3. In this direction, we assume that the coverability condition is satisfied, and we repair a generic word $u \in \mathscr{L}(\mathcal{R})$ as follows. We first consider the path π of SCCs of $\mathcal{R}$ that is induced by a successful run on u, and we observe that the input word u can be factorized into a bounded number of pieces, each one realizable within a component of the path. For instance, if $\pi = C_1 \dots C_k$ is the considered path inside $\mathsf{dag}(\mathcal{R})$, then we can factorize u as $u_1\ a_1\ u_2\ \dots\ a_{k-1}\ u_k$, with $u_i \in \mathscr{L}(\mathcal{R}|C_i)$ and $a_j \in \Sigma$ for all $1 \le i \le k$ and all $1 \le j < k$. We then consider a path inside $\mathsf{dag}^*(\mathcal{T})$ that covers π, say $\tau = D_1 \dots D_k$, and we observe that each factor u_i is also realizable within the component D_i of $\mathcal{T}$. In particular, no repair is needed on the factors $u_1, \dots, u_k$. On the other hand, the characters $a_1, \dots, a_{k-1}$ that are interleaved with the factors in u and that induce jumps along the components $C_1, \dots, C_k$ of π can be replaced by small strings inducing analogous jumps along the components $D_1, \dots, D_k$ of τ. This is possible because there exist partial runs connecting arbitrary states in each component D_i to arbitrary states in the next component D_{i+1}. In the end, the repair of u is formed by interleaving the factors $u_1, \dots, u_k$ with the small strings that replace $a_1, \dots, a_{k-1}$, and by appending a final string to reach an accepting state of $\mathcal{T}$.

We now turn to the technical details of this proof. Suppose that every path in $\mathsf{dag}(\mathcal{R})$ is covered by some path in $\mathsf{dag}^*(\mathcal{T})$. We have to prove the existence of a repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with edit cost uniformly bounded by $(1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|$. Let us fix a generic word u in the restriction language $\mathscr{L}(\mathcal{R})$, and let ρ be an accepting run of $\mathcal{R}$ on u. The sequence of SCCs of $\mathcal{R}$ that are visited by the run ρ identifies a path inside $\mathsf{dag}(\mathcal{R})$, say $\pi = C_1 C_2 \dots C_k$. Accordingly, we factorize the word u into the sequence of subwords $u_1, a_1, u_2, a_2, \dots, u_k$, where $u_i \in \mathscr{L}(\mathcal{R}|C_i)$ for all $1 \le i \le k$ and $a_i \in \Sigma$

for all $1 \le i \le k-1$. From the assumption of coverability of the paths in $\mathsf{dag}(\mathcal{R})$, we know that there is a path $\tau = D_1 D_2 \dots D_k$ in $\mathsf{dag}^*(\mathcal{T})$ that covers π, namely, such that $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|D_i)$ for all $1 \le i \le k$. This shows that each subword u_i, with $1 \le i \le k$, belongs also to the language $\mathscr{L}(\mathcal{T}|D_i)$.

We can now construct inductively a corresponding word $f(u)$ that has the form $v_0\ u_1\ v_1\ u_2\ \dots\ u_k\ v_k$ and that is accepted by $\mathcal{T}$. Recall that $\tau = D_1 \dots D_k$ is a path in $\mathsf{dag}^*(\mathcal{T})$. Because the automaton $\mathcal{T}$ is trimmed, we know that all states in D_1 can be reached from some initial state of $\mathcal{T}$ and, similarly, all states in D_k can reach some final state. In particular, since $u_1 \in \mathscr{L}(\mathcal{T}|D_1)$, we know that there exist an initial state s_0 of $\mathcal{T}$, two states r_1 and s_1 in D_1, and a word $v_0 \in \Delta^*$ such that $\mathcal{T}$ admits a run of the form $s_0 \xrightarrow{v_0} r_1 \xrightarrow{u_1} s_1$. Moreover, without loss of generality, we can assume that the length of v_0 is bounded by the number of states of $\mathcal{T}$. For the inductive step, we assume that the words $v_0, \dots, v_{i-1}$, with $0 \le i \le k-1$, are defined and that $\mathcal{T}$ admits a run on $v_0\ u_1\ \dots\ v_{i-1}$ from the initial state s_0 to a state $s_i \in D_i$. Since every state in D_{i+1} is reachable from every state in D_i, we know that there is a word v_i, with $|v_i| \le |\mathcal{T}|$, and two states r_{i+1} and s_{i+1} in D_{i+1} such that $\mathcal{T}$ admits a run of the form $s_i \xrightarrow{v_i} r_{i+1} \xrightarrow{u_{i+1}} s_{i+1}$. For the final step, we assume that $v_0, \dots, v_{k-1}$ are defined and that $\mathcal{T}$ admits a run on $v_0\ u_1\ \dots\ v_{k-1}\ u_k$ from the initial state s_0 to a state $s_k \in D_k$. Using arguments similar to the previous ones and the fact that all states in D_k can reach a final state, we derive the existence of a word v_k, with $|v_k| \le |\mathcal{T}|$, and a final state r_{k+1} of $\mathcal{T}$ such that $\mathcal{T}$ admits a run of the form $s_k \xrightarrow{v_k} r_{k+1}$. Putting everything together, we obtain the existence of a successful run of $\mathcal{T}$ of the form

$$s_0 \xrightarrow{v_0} r_1 \xrightarrow{u_1} s_1 \xrightarrow{v_1} r_2 \xrightarrow{u_2} \dots\ r_k \xrightarrow{u_k} s_k \xrightarrow{v_k} r_{k+1}$$

and hence the word $f(u) = v_0\ u_1\ v_1\ u_2\ \dots\ u_k\ v_k$ is accepted by $\mathcal{T}$. Moreover, since $k \le |\mathsf{dag}(\mathcal{R})|$ and $|v_i| \le |\mathcal{T}|$ for all $0 \le i \le k$, we have that the edit distance between u and $f(u)$ is at most $(1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|$. This proves that there is a repair strategy f for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ with edit cost uniformly bounded by $(1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|$, and hence

$$\mathsf{cost}(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})) \ \le\ (1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|.$$

We now prove the implication from 1 to 2, namely, we assume that there is a repair strategy f for $\mathcal{R}$ and $\mathcal{T}$ with bounded cost and we show that every path in $\mathsf{dag}(\mathcal{R})$ is covered by a path in $\mathsf{dag}^*(\mathcal{T})$. The general idea is to associate with any path $\pi = C_1, \dots, C_k$ inside $\mathsf{dag}(\mathcal{R})$ a suitable word $u_\pi \in \mathscr{L}(\mathcal{R})$, called *witnessing word* of π, whose repair according to the strategy f induces a path τ inside $\mathsf{dag}^*(\mathcal{T})$ that covers π. Intuitively, the witnessing word u_π is obtained from the path π by replacing every component C_i with a sufficiently large number of repetitions of a special string in $\mathscr{L}(\mathcal{R}|C_i)$, which we called *fingerprint of* C_i. The number of repetitions of each fingerprint will depend on the worst-case repair cost $N = \max_{u \in \mathscr{L}(\mathcal{R})}\{\mathsf{cost}(u, f)\}$, and will be large enough to imply that any repair of the witnessing word u_π achieved by at most N edits contains at least one copy of the fingerprint of each component C_i of π. Moreover, the order

of occurrence of the fingerprints inside the repaired string will be the same as the order of the components in the path π. One finally looks at some run of $\mathcal{T}$ that accepts $\mathsf{f}(u_\pi)$: thanks to the occurrence order of the fingerprints of $\mathsf{f}(u_\pi)$, this run must induce a path τ inside $\mathsf{dag}^*(\mathcal{T})$ that covers π.

Before constructing the witnessing word, we prove a technical lemma, which defines precisely the concept of fingerprint of a component of $\mathcal{R}$. Intuitively, the lemma shows that, for any given component C of $\mathcal{R}$, one can find a word u that can be arbitrarily "pumped" inside the language $\mathscr{L}(\mathcal{R}|C)$ – namely, $\{u\}^* \subseteq \mathscr{L}(\mathcal{R}|C)$ – and such that, for all components D of $\mathcal{T}$, $u \in \mathscr{L}(\mathcal{T}|D)$ iff $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D)$. Hereafter, we say that a word u is *cyclic* (in a component C) if there is $q \in C$ such that $\mathcal{R}$ admits a run of the form $q \xrightarrow{u} q$.

Lemma 4.2. *For every component C of $\mathcal{R}$, there is a* cyclic *word $u \in \mathscr{L}(\mathcal{R}|C)$ such that, for every component D of $\mathcal{T}$,*

$$u \in \mathscr{L}(\mathcal{T}|D) \qquad \textit{iff} \qquad \mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D).$$

Proof. Let C be a component of $\mathcal{R}$ and let $D_1, \ldots, D_m$ be all the components of $\mathcal{T}$. We construct the cyclic word u by exploiting an induction over the number of components in $\mathcal{T}$. That is, we prove that for all i, there is a cyclic word $u_i \in \mathscr{L}(\mathcal{R}|C)$ such that:

$$\forall 1 \leq j \leq i. \qquad \mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D_j) \quad \text{iff} \quad u_i \in \mathscr{L}(\mathcal{T}|D_j) \qquad (\star)$$

Clearly, the lemma follows from $(\star)$ when we let $u = u_m$.

The base step $i = 0$ is vacuously true, so we simply let $u_0 = \varepsilon$. For the inductive step, let $i < m$ and suppose that there exists a word u_i satisfying $(\star)$. We construct a word u_{i+1} that satisfies $(\star)$ as well. We distinguish between two cases, depending on whether $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D_{i+1})$ or not. If $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D_{i+1})$, then we simply let $u_{i+1} = u_i$, in such a way that $(\star)$ holds trivially. Otherwise, let v_1 be a word in $\mathscr{L}(\mathcal{R}|C) \setminus \mathscr{L}(\mathcal{T}|D_{i+1})$. Since $v_1 \in \mathscr{L}(\mathcal{R}|C)$, $\mathcal{R}$ admits a run of the form $q \xrightarrow{v_1} r$, for some states $q, r \in C$. Similarly, since u_i is cyclic in C, $\mathcal{R}$ admits a run of the form $s \xrightarrow{u_i} s$, for some state $s \in C$. Moreover, since q, r, s are mutually reachable inside C, there exist some words $v_0, v_2 \in \mathscr{L}(\mathcal{R}|C)$ such that $\mathcal{R}$ admits runs of the form $s \xrightarrow{v_0} q$ and $r \xrightarrow{v_2} s$. Now, define $u_{i+1} = v_0\, v_1\, v_2\, u_i$. The word u_{i+1} is cyclic in C, as witnessed by following the run:

$$s \xrightarrow{v_0} q \xrightarrow{v_1} r \xrightarrow{v_2} s \xrightarrow{u_i} s.$$

It is also easy to see that u_{i+1} does not belong to $\mathscr{L}(\mathcal{T}|D_{i+1})$. Indeed, if $u_{i+1} \in \mathscr{L}(\mathcal{T}|D_{i+1})$, then the subautomaton $\mathcal{T}|D_{i+1}$ would also admit a run on the factor v_1 of u_{i+1}, which is a contradiction since $v_1 \notin \mathscr{L}(\mathcal{T}|D_{i+1})$. Finally, we know from the inductive hypothesis that for all $1 \leq j \leq i$, $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D_j)$ iff $u_i \in \mathscr{L}(\mathcal{T}|D_j)$, and that the latter holds iff $u_{i+1} \in \mathscr{L}(\mathcal{T}|D_j)$. We conclude that, for every $1 \leq j \leq i+1$, $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D_j)$ iff $u_{i+1} \in \mathscr{L}(\mathcal{T}|D_j)$, which proves the inductive step for $(\star)$. □

Consider now any path $\pi = C_1 \dots C_k \in \mathsf{dag}(\mathcal{R})$. We associate with each component C_i a word u_i that satisfies Lemma 4.2, and we call it *the fingerprint of* C_i. Since each word u_i is cyclic for C_i and since all states in a component C_i can reach all states in the next component C_{i+1}, we know that there exist states $q_1 \in C_1, \dots, q_k \in C_k$ and words $u'_1, \dots, u'_{k-1}$ such that $\mathcal{R}$ admits a run of the form

$$q_1 \xrightarrow{u_1} q_1 \xrightarrow{u'_1} q_2 \xrightarrow{u_2} q_2 \xrightarrow{u'_2} \cdots \xrightarrow{u_{k-1}} q_k \xrightarrow{u_k} q_k.$$

Furthermore, since $\mathcal{R}$ is trimmed, there exist an initial state q_0, a final state q_{k+1}, and some words u'_0, u'_k such that $\mathcal{R}$ admits runs of the form $q_0 \xrightarrow{u'_0} q_1$ and $q_k \xrightarrow{u'_k} q_{k+1}$. Putting all together, we have that, for every positive natural number n, the word

$$u^{(n)} \;=^{\text{def}}\; u'_0\, u_1^n\, u'_1\, u_2^n \;\dots\; u_k^n\, u'_k$$

is accepted by $\mathcal{R}$. We define the witnessing word of π as $u_\pi = u^{(M)}$, where $M = (N+1) \cdot (|\mathsf{dag}(\mathcal{R})| + 1)$.

To conclude the proof, we look at the repair $f(u_\pi)$ of the witnessing word of π, we extract from it a path τ in $\mathsf{dag}^*(\mathcal{T})$, and we show that τ covers π. First, recall that $f(u_\pi)$ is accepted by $\mathcal{T}$ and is obtained from u_π by a sequence of at most N edits. From this and the definition of u_π, it follows that we can write

$$f(u_\pi) \;=\; v_0\, u_1^\ell\, v_1\, u_2^\ell \;\dots\; u_k^\ell\, v_k$$

where $\ell = |\mathsf{dag}(\mathcal{R})| + 1$. A simple application of the pigeon-hole principle shows that every run of $\mathcal{T}$ on $f(u_\pi)$ must contain a sub-run of the form $r_i \xrightarrow{u_i} s_i$, for each $1 \le i \le k$, where the states r_i and s_i belong to the same component of $\mathcal{T}$. Specifically, this means that $f(u_\pi)$ can be equally written as $v'_0\, u_1\, v'_1\, u_2 \;\dots\; u_k\, v'_k$, and that this word is accepted by a run of $\mathcal{T}$ of the form:

$$s_0 \xrightarrow{v'_0} r_1 \xrightarrow{u_1} s_1 \xrightarrow{v'_1} r_2 \xrightarrow{u_2} s_2 \;\dots\; r_k \xrightarrow{u_k} s_k \xrightarrow{v'_k} r_{k+1}$$

where $r_i, s_i \in D_i$ for all $1 \le i \le k$, and $D_1, \dots, D_k$ are SCCs of $\mathcal{T}$. In particular, for all $1 \le i \le k$, we have $u_i \in \mathscr{L}(\mathcal{T}|D_i)$, and hence $\mathscr{L}(\mathcal{R}|C_i) \subseteq \mathscr{L}(\mathcal{T}|D_i)$ since u_i is the fingerprint of C_i. We have just shown that the path $\tau = D_1 \dots D_k$ in $\mathsf{dag}^*(\mathcal{T})$ covers the arbitrarily chosen path $\pi = C_1 \dots C_k$. □

4.2. Streaming setting

We now provide a variant of Theorem 4.1 that characterizes the positive instances of the bounded repair problem in the *streaming setting*. We do so by adding in a game. For this it is convenient to assume that the restriction and target languages are presented by means of DFA $\mathcal{R}$ and $\mathcal{T}$. We associate with $\mathcal{R}$ and $\mathcal{T}$ a *reachability game* played by two players, Adam and Eve, on a directed acyclic graph $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, which is defined in terms of the SCCs of $\mathcal{R}$ and $\mathcal{T}$. The idea underlying the game is as follows: the moves of Adam's construct incrementally a path π in $\mathsf{dag}(\mathcal{R})$; Eve has to respond to these moves by constructing of a corresponding path $f(\pi)$ in $\mathsf{dag}^*(\mathcal{T})$ that covers π. In the game setting, moves are played alternatively between Adam and Eve. In particular, the function f

that describes Eve's strategy must satisfy the following condition: if $\pi \cdot \mathsf{C}$ is an extension of the path π in $\mathsf{dag}(\mathcal{R})$ by a single component, then either $\mathsf{f}(\pi \cdot \mathsf{C})$ coincides with $\mathsf{f}(\pi)$ or it is an extension of $\mathsf{f}(\pi)$ by a single component, namely, $\mathsf{f}(\pi \cdot \mathsf{C})$ is of the form $\mathsf{f}(\pi) \cdot \mathsf{D}$.

Formally, the *arena* $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ of the game is a directed acyclic graph, in which the nodes owned by Adam (resp., Eve) are the pairs of the form (C, D) (resp., (D, C)), with C SCC of $\mathcal{R}$ and D SCC of $\mathcal{T}$. The edges of the arena connect Adam's nodes (C, D) to Eve's nodes $(\mathsf{D}, \mathsf{C}')$ whenever $(\mathsf{C}, \mathsf{C}')$ is an edge of $\mathsf{dag}(\mathcal{R})$; similarly, they connect Eve's nodes (D, C) to Adam's nodes $(\mathsf{C}, \mathsf{D}')$ whenever $(\mathsf{D}, \mathsf{D}')$ is an edge of $\mathsf{dag}^*(\mathcal{T})$ and, in addition, $\mathscr{L}(\mathcal{R}|\mathsf{C}) \subseteq \mathscr{L}(\mathcal{T}|\mathsf{D}')$. The initial node is owned by Eve and it is of the form $(\mathsf{D}_0, \mathsf{C}_0)$, 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 the choices of the other player.

The following characterization reduces the bounded repair problem in the streaming setting to the problem of determining the winner of a reachability game over a finite arena.

Theorem 4.3. *Given two DFA $\mathcal{R}$ and $\mathcal{T}$, the following two conditions are equivalent:*

1. *there exists a streaming strategy with some lookahead that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded edit cost,*
2. *Eve has a winning strategy in the reachability game on $\mathcal{A}_{\mathcal{R},\mathcal{T}}$.*

The idea underlying the proof of the direction from 2 to 1 is as follows. If we have a winning strategy for Eve, then we can get a streaming repair strategy for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ by tracking the current component C reached by the input and by maintaining the invariant that the component D induced by repaired string is such that (C, D) is a position consistent with Eve's winning strategy. When a new letter comes in and changes the component in the restriction from C to C', we respond with a suitable repair that moves from D to the response component D' that preserves the invariant.

For the direction from 1 to 2, we assume that there is a streaming k-lookahead edit strategy that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost and derive a strategy for Eve. The strategy will maintain the invariant that the position (C, D) corresponds to some input string $\mathfrak{u}$ and repair $\mathfrak{v}$ consistent with the repair strategy. If, by way of contradiction, we reach a pair (C, D) corresponding to some string $\mathfrak{u}$ and, moreover, there is a successor SCC C' of C corresponding to some extension $\mathfrak{u}\mathfrak{u}'$, and $(\mathsf{D}, \mathsf{C}')$ is a losing position for Eve, then, for every SCC D' of $\mathcal{T}$, we can find a word witnessing the non-containment $\mathscr{L}(\mathcal{R}|\mathsf{C}') \not\subseteq \mathscr{L}(\mathcal{T}|\mathsf{D}')$. We can then concatenate multiple copies of such words together in such a way that the resulting string cannot be re-

paired by our transducer with a bounded number of edit operations, which is a contradiction.

Before turning to the proof of the above theorem, it is convenient to establish two preliminary lemmas, which will be also reused in other proofs (e.g., in the proof of Proposition 5.4). 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 4.4. *Let $\mathcal{R}$ be an NFA, let $\mathcal{T}$ be a DFA, and let C and D be some SCCs of $\mathcal{R}$ and $\mathcal{T}$, respectively. We have that*

$$\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D) \qquad \textit{iff} \qquad \exists\, q \in C,\ r \in D.\ \ \mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_r D).$$

Proof. Let $\mathcal{R} = (\Sigma, Q, E, I, F)$ and $\mathcal{T} = (\Delta, Q', \delta', r_0, F')$. We first prove the right-to-left implication. Suppose that $\mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_r D)$ for some states $q \in C$ and $r \in D$. 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 $u_0\, u \in \mathscr{L}(\mathcal{R}|_q C)$. Since $\mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_r D)$, the same word $u_0\, u$ belongs also to the language $\mathscr{L}(\mathcal{T}|_r D)$. In particular, this shows that $u \in \mathscr{L}(\mathcal{T}|D)$.

We now prove the contrapositive of the left-to-right implication, namely, we assume that $\mathscr{L}(\mathcal{R}|_q C) \nsubseteq \mathscr{L}(\mathcal{T}|_r D)$ for all $q \in C$ and all $r \in D$ and we derive $\mathscr{L}(\mathcal{R}|C) \nsubseteq \mathscr{L}(\mathcal{T}|D)$. Let $D = \{r_1, \ldots, r_n\}$. We first prove, by exploiting an induction on $i \leq n$, that there is a word u_i that belongs to $\mathscr{L}(\mathcal{R}|C)$ but not to $\mathscr{L}(\mathcal{T}|_{r_j} D)$ for all indices $1 \leq j \leq i$. The base case is trivial. For the inductive step, we assume that the claim holds for $i < n$ and we prove it for $i + 1$. The main idea is to append to the word obtained from the inductive hypothesis a suitable word that witnesses a non-containment of $\mathscr{L}(\mathcal{R}|_q C)$ into $\mathscr{L}(\mathcal{T}|_{r_{i+1}} D)$. Let u_i be the word obtained from the inductive hypothesis such that $u_i \in \mathscr{L}(\mathcal{R}|C) \setminus \mathscr{L}(\mathcal{T}|_{r_j} D)$ for all $1 \leq j \leq i$. Moreover, let s_{i+1} be the state reached by $\mathcal{T}$ from r_{i+1} after parsing the word u_i, that is, $s_{i+1} = \delta'(r_{i+1}, u_i)$ (note that here we use the determinism of $\mathcal{T}$). If $s_{i+1} \notin D$, then the claim follows trivially. We thus consider the case $s_{i+1} \in D$. Recall that $u_i \in \mathscr{L}(\mathcal{R}|C)$ and let p_i, p'_i be two states in C such that $p_i \xrightarrow{u_i} p'_i$. From the original assumption and the fact that $p'_i \in C$ and $s_{i+1} \in D$, we know that $\mathscr{L}(\mathcal{R}|_{p'_i} C) \nsubseteq \mathscr{L}(\mathcal{T}|_{s_{i+1}} D)$, and hence there is a word $v_{i+1} \in \mathscr{L}(\mathcal{R}|_{p'_i} C) \setminus \mathscr{L}(\mathcal{T}|_{s_{i+1}} D)$. In particular, this shows that the word $u_{i+1} = u_i\, v_{i+1}$ belongs to the language $\mathscr{L}(\mathcal{R}|C)$, but not to the language $\mathscr{L}(\mathcal{T}|_{r_{i+1}} D)$. Finally, since D is a SCC, it follows that u_{i+1} does not belong to any of the languages $\mathscr{L}(\mathcal{T}|_{r_j} D)$ either, for all $1 \leq j \leq i$, and this concludes the proof. □

Lemma 4.5. *Let $\mathcal{R}$ and $\mathcal{T}$ be two DFA and let C and D be some SCCs of $\mathcal{R}$ and $\mathcal{T}$, respectively. We have that*

$$\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D) \qquad \textit{implies} \qquad \forall\, q \in C.\ \exists\, r \in D.\ \ \mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_r D).$$

Proof. Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', r_0, F')$ be two DFA, let C and D be two SCC of $\mathcal{R}$ and $\mathcal{T}$, respectively, such that $\mathscr{L}(\mathcal{R}|C) \subseteq \mathscr{L}(\mathcal{T}|D)$, and

let q be a state in C. We know from Lemma 4.4 that there exist two states $p \in C$ and $s \in D$ such that $\mathscr{L}(\mathcal{R}|_p C) \subseteq \mathscr{L}(\mathcal{T}|_s D)$. Moreover, since the states p, q in C are mutually reachable, there is a word u_0 such that $\delta(p, u_0) = q$. Since $\mathscr{L}(\mathcal{R}|_p C) \subseteq \mathscr{L}(\mathcal{T}|_s D)$ and $u_0 \in \mathscr{L}(\mathcal{R}|_p C)$, we know that the state $r = \delta(s, u_0)$ belongs to D as well. Now, let us consider a generic word u in $\mathscr{L}(\mathcal{R}|_q C)$. Clearly, we have $u_0\, u \in \mathscr{L}(\mathcal{R}|_p C)$. From $\mathscr{L}(\mathcal{R}|_p C) \subseteq \mathscr{L}(\mathcal{T}|_s D)$ it follows that $u_0\, u \in \mathscr{L}(\mathcal{T}|_s D)$, whence $u \in \mathscr{L}(\mathcal{T}|_r D)$. This shows that $\mathscr{L}(\mathcal{R}|_q C) \subseteq \mathscr{L}(\mathcal{T}|_r D)$. □

Proof of Theorem 4.3. Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', r_0, F')$ be two DFA and let $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ be the arena obtained from $\mathcal{R}$ and $\mathcal{T}$ as described above. Below, we prove first the implication from 2 to 1 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 $C \in \mathsf{dag}(\mathcal{R})$ and $D \in \mathsf{dag}^*(\mathcal{T})$, 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 subsequential 0-lookahead transducer $\mathcal{S}$ that implements a streaming repair strategy for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ having uniformly bounded aggregate cost. Intuitively, the transducer $\mathcal{S}$ works as follows. While parsing the input word u from the restriction language $\mathscr{L}(\mathcal{R})$ and emitting a corresponding word v, the transducer $\mathcal{S}$ mimics, at the same time, the transitions of both automata $\mathcal{R}$ and $\mathcal{T}$. Each time the restriction automaton $\mathcal{R}$ exits the current SCC C and enters a new SCC C', a corresponding move $(C, D) \xrightarrow{\mathsf{Adam}} (D, C')$ 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 4.5 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 duplicating states). Then, we let C_0 be the SCC of q_0 in $\mathcal{R}$, D_0 be the SCC of the initial state r_0 of $\mathcal{T}$, $g(D_0, C_0) = (C_0, D_1)$ 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). We also let r_1 be some arbitrary state in D_1 such that $\mathscr{L}(\mathcal{R}|_{q_0} C_0) \subseteq \mathscr{L}(\mathcal{T}|_{r_1} D_1)$ – this states exists thanks to the containment $\mathscr{L}(\mathcal{R}|C_0) \subseteq \mathscr{L}(\mathcal{R}|D_1)$ and Lemma 4.5. Accordingly, we define v_0 to be the shortest word such that $\delta'(r_0, v_0) = r_1$ – note that $\mathsf{dag}^*(\mathcal{T})$ contains an edge from D_0 to D_1 and hence the state r_1 must be reachable from r_0. Intuitively, the word v_0 is the first prefix emitted by the transducer $\mathcal{S}$ at the beginning of

[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.

the computation (it should be now clear why we assumed that the initial state q_0 of $\mathcal{R}$ has no entering transitions). Below we define the various components of the transducer $\mathcal{S} = (\Sigma, \Delta, Q'', \delta'', s_0, \Omega)$.

- The state space of the transducer is $Q'' = Q \times Q'$.
- For every state $s = (q, r) \in Q''$ and every symbol $a \in \Sigma$, $\delta''(s, a) = (v, s')$, where $s' = (\delta(q, a), \delta'(r, v))$ and v is the word over Δ defined as follows:
  1. if both states q and $\delta(q, a)$ 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 $q \neq q_0$ or $q = q_0$;
  2. otherwise, if the states q and $\delta(q, a)$ belong to different SCCs, we denote by C' the SCC of the state $\delta(q, a)$ in $\mathcal{R}$, by D the SCC of the state r in $\mathcal{T}$, by (C', D') the response $g(D, C')$ given by Eve's winning strategy, by r' some arbitrary state in D' such that $\mathscr{L}(\mathcal{R}|_{\delta(q,a)} C') \subseteq \mathscr{L}(\mathcal{T}|_{r'} D')$ – the state r' exists thanks to the containment $\mathscr{L}(\mathcal{R}|C') \subseteq \mathscr{L}(\mathcal{R}|D')$ and Lemma 4.5 – and, finally, we denote by v' the shortest word such that $\delta'(r, v') = r'$ – the state r' is reachable from r because $\mathsf{dag}^*(\mathcal{T})$ contains an edge from D to D'. Accordingly, we define the output v be either the word v' or the word $v_0\, v'$, depending on whether $q \neq q_0$ or $q = q_0$.
- The initial state of the transducer is $s_0 = (q_0, r_0)$.
- For every state $s = (q, r) \in Q''$, the final output $\Omega(s)$ is defined to be the shortest word $v \in \Delta^*$ such that $\delta(r, v) \in F'$ – 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.

Using arguments similar to those used in the proof of Theorem 4.1 (e.g., by associating with each successful run of $\mathcal{R}$ a corresponding path π in $\mathsf{dag}(\mathcal{R})$ and a covering path τ in $\mathsf{dag}^*(\mathcal{T})$ 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 aggregate cost of $\mathcal{S}$ on input $u \in \mathscr{L}(\mathcal{R})$ is at most $(1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|$ and hence

$$\mathsf{cost}^{\mathsf{aggr}}_{0-\mathsf{lookahead}}(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})) \;\leq\; (1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|.$$

We now prove the implication from 1 to 2, namely, we assume that $\mathcal{S} = (\Sigma, \Delta, Q'', \delta'', s_0, \Omega)$ is a transducer with k-lookahead that implements a repair strategy for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ with edit cost uniformly bounded by a natural number N, and we derive from this a winning strategy for Eve. More precisely, we will specify Eve's moves $g(D, C)$ on those nodes (D, C) in $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, with $C \in \mathsf{dag}(\mathcal{R})$ and $D \in \mathsf{dag}^*(\mathcal{T})$, for which there exist some words $u_C, u'_C \in \Sigma^*$ and $v_D \in \Delta^*$ such that

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

ii) $\delta'(r_0, v_D) \in D$,

iii) $\delta''(s_0, u_C) = (v_D, s')$ for some $s' \in Q''$.

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 the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, then $g(D', C')$ is defined as well. Thus, to prove that the defined strategy g is winning for Eve, it will be sufficient to verify that the moves induced at the leaves of g reach nodes that are losing for Adam. Below, we define the moves $g(D, C)$ by exploiting an induction on the distance of the SCC C from the root of $\mathsf{dag}(\mathcal{R})$, which is the SCC of the initial state q_0 of $\mathcal{R}$.

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) $\delta(q_0, u_C\ u'_C) \in C$, (ii) $\delta'(r_0, v_D) \in D$, and (iii) $\delta''(s_0, u_C) = (v_D, s')$ for some $s' \in Q''$. For the sake of brevity, we let $q = \delta(q_0, u_C\ u'_C)$. We claim that there is a SCC D' that is a successor of D in $\mathsf{dag}^*(\mathcal{T})$ (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 4.1, we can recursively construct a sequence of words $u'_0, u_1, u'_1, \ldots, u_h, u'_h$ over Σ such that, for all $1 \leq i \leq h$:

i) $q \xrightarrow{u_i\, u'_i} q$ is a run of $\mathcal{R}$ that starts and ends in state q,

ii) $u_i \notin \mathscr{L}(\mathcal{T}|D_i)$.

In particular, the first property implies that the word

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

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 $\mathcal{S}(u)$, $\delta(r_0, v_D) \in D$, and $u_i \notin \mathscr{L}(\mathcal{T}|D_i)$ for all $1 \leq i \leq h$, we have that $\mathcal{S}(u)$ cannot belong to the target language $\mathscr{L}(\mathcal{T})$. This is against the assumption that $\mathcal{S}$ implements a repair strategy for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$. We must conclude that there is a SCC D' that is a successor of D in $\mathsf{dag}^*(\mathcal{T})$ 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, in every partial

play of the form

$$(D_0, C_0) \xrightarrow{\mathsf{Eve}} (C_0, D_1) \xrightarrow{\mathsf{Adam}} \dots \xrightarrow{\mathsf{Adam}} (D_n, C_n)$$

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). To prove this property it is sufficient to check that the node (D_n, C_n) is reachable: this can be easily verified by exploiting an induction on n and the definition of g given above. We conclude that Eve has a winning strategy for the reachability game over $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. □

It is interesting to observe that the above proof also shows that the property of bounded repairability in the streaming setting is not sensitive to the notions of transducer with/without lookahead, nor to the models of aggregate/edit cost:

Corollary 4.6. *For two DFA $\mathcal{R}, \mathcal{T}$, the following two conditions are equivalent:*

1. *there exist $k \in \mathbb{N}$ and a streaming strategy* with k-lookahead *that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with* edit cost *uniformly bounded by a constant,*
2. *there exists a streaming strategy* with 0-lookahead *that repairs $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with* aggregate cost *uniformly bounded by $(1 + |\mathsf{dag}(\mathcal{R})|) \cdot |\mathcal{T}|$.*

5. Complexity analysis

In this section we analyse the complexity of the bounded repair problem and the threshold problem, distinguishing, in both cases, between the non-streaming and streaming settings and between NFA- and DFA-based specifications of regular languages. We will also consider special cases where the restriction or the target languages are fixed, and when the restriction language is universal (i.e. no restriction).

5.1. The bounded repair problem in the non-streaming setting

NFA vs NFA. Theorem 4.1 leads straightforwardly to a polynomial-space algorithm that solves the bounded repair problem between two NFA $\mathcal{R}$ and $\mathcal{T}$ in the non-streaming setting: the algorithm first guesses universally a path $\pi = C_1 \dots C_k$ in $\mathsf{dag}(\mathcal{R})$, then it guesses existentially a path $\tau = D_1 \dots D_k$ of the same length in $\mathsf{dag}^*(\mathcal{T})$, and finally it checks the containment of the sub-automaton $\mathcal{R}|C_i$ in the sub-automaton $\mathcal{T}|D_i$ for all indices $1 \le i \le n$ (these containments can be checked in polynomial space [6]). Together with the PSPACE lower bound proved later in Corollary 5.10 (which deals with the unrestricted case), we obtain:

Corollary 5.1. *The bounded repair problem in the non-streaming setting, where both the restriction and target languages are specified by NFA, is PSPACE-complete.*

NFA vs DFA. The same characterization result can be used to solve the bounded repair problem when the restriction language is specified by an NFA and the target language is specified by a DFA. In this case, we can take advantage of the determinism and show that the problem becomes coNP-complete:

Theorem 5.2. *The bounded repair problem in the non-streaming setting, where the restriction language is specified by an NFA and the target language is specified by a DFA, is in coNP and it is coNP-hard already for languages specified by DFA.*

Before turning to the proof of the above theorem, we establish the following complexity result for the coverability problem:

Lemma 5.3. *Given two NFA $\mathcal{R}$ and $\mathcal{T}$ and a path $\pi = \mathsf{C}_1 \dots \mathsf{C}_k$ in $\mathsf{dag}(\mathcal{R})$, the problem of deciding whether π is covered by some path in $\mathsf{dag}^*(\mathcal{T})$ is in PTIME with an oracle for deciding containment of the languages $\mathscr{L}(\mathcal{R}|\mathsf{C}_i)$ inside language of the form $\mathscr{L}(\mathcal{T}|\mathsf{D})$, with D SCC of $\mathcal{T}$.*

Proof. The basic idea for checking whether the path π is covered by some path in $\mathsf{dag}^*(\mathcal{T})$ is to incrementally process longer and longer prefixes of π while keeping the frontier of the paths in $\mathsf{dag}^*(\mathcal{T})$ that cover these prefixes. Algorithm 5.1 below gives the pseudo-code for a procedure that implements this idea:

Algorithm 5.1: PathCoverability($\mathcal{R}$, $\mathcal{T}$, π)

```
let π = C_1 ... C_k
let dag*(T) = (V', E')
F ← ∅
for all D ∈ V'
  do { if CheckContainment(R, T, C_1, D)
       then F ← F ∪ {D}
for i ← 2 to k
  do { F' ← F
       F ← ∅
       for all D' ∈ F' and (D', D) ∈ E'
         do { if CheckContainment(R, T, C_i, D)
              then F ← F ∪ {D}
return (F ≠ ∅)
```

Note that the pseudo-code uses CheckContainment$(\mathcal{R}, \mathcal{T}, \mathsf{C}_i, \mathsf{D})$ as an oracle for deciding containment between the languages $\mathscr{L}(\mathcal{R}|\mathsf{C}_i)$ and $\mathscr{L}(\mathcal{T}|\mathsf{D})$. The proof of the correctness of the algorithm is based on the following invariant: at each iteration of the loop on i, the set F contains a SCC D of $\mathcal{T}$ iff there is a path $\tau = \mathsf{D}_1 \dots \mathsf{D}_i$ in $\mathsf{dag}^*(\mathcal{T})$, with $\mathsf{D}_i = \mathsf{D}$, that covers $\pi_i = \mathsf{C}_1 \dots \mathsf{C}_i$. The described procedure runs in polynomial time with respect to the size of the input NFA $\mathcal{R}$ and $\mathcal{T}$. □

Proof of Theorem 5.2. We first prove the complexity upper bound. Let $\mathcal{R}$ be an NFA and $\mathcal{T}$ be a DFA. Thanks to Theorem 4.1, deciding whether $\mathsf{cost}(\mathscr{L}(\mathcal{R}), \mathscr{L}(\mathcal{T})) < \infty$ amounts at first guessing universally a path π in $\mathsf{dag}(\mathcal{R})$ (this can be done in coNP) and then checking whether π is covered by some path in $\mathsf{dag}^*(\mathcal{T})$. By Lemma 5.3, this can be done efficiently using an oracle for checking containment of languages recognized by SCCs of $\mathcal{R}$ and $\mathcal{T}$.

What remains to be done is to show that one can decide in polynomial time the containment of any language $\mathscr{L}(\mathcal{R}|\mathsf{C})$ inside any language $\mathscr{L}(\mathcal{T}|\mathsf{D})$, where C (resp., D) is a SCC of the NFA $\mathcal{R}$ (resp., DFA $\mathcal{T}$) – note that, strictly speaking, the sub-automaton $\mathcal{T}|\mathsf{D}$ is not deterministic, since its successful runs can start from arbitrary states in D. To show this we use Lemma 4.4, which reduces the containment problem $\mathscr{L}(\mathcal{R}|\mathsf{C}) \subseteq \mathscr{L}(\mathcal{T}|\mathsf{D})$ to a series of containment problems between a non-deterministic sub-automaton of $\mathcal{R}$ (i.e., $\mathcal{R}|_q\mathsf{C}$ for some $q \in \mathsf{C}$) and a *deterministic* sub-automaton of $\mathcal{T}$ (i.e., $\mathcal{T}|_r\mathsf{D}$ for some $r \in \mathsf{D}$). Since deterministic automata can be easily complemented, the latter instances of the containment problem can be decided in polynomial time by reducing to an emptiness problem involving the intersection of two automata (i.e., $\mathcal{R}|_q\mathsf{C}$ and the complement of $\mathcal{T}|_r\mathsf{D}$).

Putting all together, we have that the bounded repair problem in the non-streaming setting, where the restriction language is given by an NFA and the target language is given by a DFA, is in coNP.

We now prove the lower bound, which follows from a reduction from the validity problem for propositional formulas in disjunctive normal form (i.e., the dual of the SAT problem). The general idea is to encode in the restriction language all the possible valuations for the propositional variables and then restrict the target language to consist only of encodings of valuations that satisfy at least one clause of the formula. We further allow some redundancy in the encodings of the valuations in order to forbid the repair strategy from modifying the encoded valuations.

We consider a set $\mathsf{X} = \{x_1, \ldots, x_k\}$ of propositional variables and we denote by $\mathsf{L} = \{x_1, \ldots, x_k\} \cup \{\neg x_1, \ldots, \neg x_k\}$ 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 $l \in \mathsf{L}$, we have $l \in \mathsf{V}$ iff $\neg l \notin \mathsf{V}$. Let us 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 $l_{i,j} \in \mathsf{L}$ for every pair of indices $1 \le i \le m$ and $1 \le j \le h_i$. 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 $1 \le i \le m$ such that $l_{i,1}, \ldots, l_{i,h_i} \in \mathsf{V}$.

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

$$\mathsf{R} \;=^{\text{def}}\; (\{x_1\}^* \cup \{\neg x_1\}^*)\ (\{x_2\}^* \cup \{\neg x_2\}^*)\ \ldots\ (\{x_k\}^* \cup \{\neg x_k\}^*).$$

Note that it is easy to construct a DFA $\mathcal{R}$ that recognizes R and that has size polynomial 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

$$T =^{\text{def}} \bigcup_{1 \le i \le m} \{a_i\} \, (L \setminus \{\neg l_{i,1}, \ \ldots, \ \neg l_{i,h_i}\})^*.$$

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. For the right-to-left implication, suppose that φ is valid and let u be a word in R. Clearly, u is 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 $1 \le i \le k$. Such a word encodes the valuation $V_u = \{l_1, l_2, \ldots, l_k\}$. Moreover, since φ is valid, there is an index $1 \le i \le m$ such that $l_{i,1}, \ldots, l_{i,h_i} \in V$. The repair strategy f for R and T could then map the word u to the word $f(u) = a_i \, u$, 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 $l_i \in \{x_i, \neg x_i\}$ for all $1 \le i \le k$, such that for every index $1 \le i \le m$, there is an index $1 \le j \le h_i$ satisfying $l_{i,j} \notin V$. We then consider the family of words $u_n = (l_1)^n \, (l_2)^n \, \ldots \, (l_k)^n$. We know that for every $1 \le i \le m$, there is $1 \le j \le h_i$ such that the edit distance between the sub-word $(\neg l_{i,j})^n$ of u_n and any word in the sub-language $T_i = \{a_i\} \, (L \setminus \{\neg l_{i,1}, \ \ldots, \ \neg l_{i,h_i}\})^*$ of T is at least n. This shows that all repair strategies of R into T have unbounded cost. □

Fixed restriction or target. Here, we briefly outline some parametrized complexity results. Quite surprisingly, the bounded repair problem in the non-streaming setting becomes tractable when we fix either the restriction language or the target language.

Proposition 5.4. *Let R be a fixed restriction language. The problem of deciding, given a DFA $\mathcal{T}$, whether $\mathsf{cost}(R, \mathscr{L}(\mathcal{T})) < \infty$ is in PTIME.*

Proof. The proof is similar to the part of the proof of Theorem 5.2 related to the coNP upper bound. Let R be a fixed restriction language and let $\mathcal{R}$ be a DFA that recognizes R. Moreover, let $\mathcal{T}$ be a given DFA recognizing the target language T. From Theorem 4.1, deciding whether $\mathsf{cost}(R, \mathscr{L}(\mathcal{T})) < \infty$ amounts at checking that every path π in $\mathsf{dag}(\mathcal{R})$ is covered by some path in $\mathsf{dag}^*(\mathcal{T})$. In virtue of Lemma 5.3 and Lemma 4.4, coverability of a given path in $\mathsf{dag}(\mathcal{R})$ by paths in $\mathsf{dag}^*(\mathcal{T})$ can be decided in polynomial time. Since the number of paths in $\mathsf{dag}(\mathcal{R})$ is fixed, this shows that the problem is in PTIME. □

Proposition 5.5. *Let T be a fixed target language. The problem of deciding, given an NFA $\mathcal{R}$, whether $\mathsf{cost}(\mathscr{L}(\mathcal{R}), T) < \infty$ is in PTIME.*

Proof. The polynomial-time solution to the bounded repair problem under a fixed target language $T = \mathscr{L}(\mathcal{T})$ uses a dynamic programming approach and the characterization of Theorem 4.1 (hence it is similar to the proof of Theorem 5.2). However, instead of guessing a path π in $\mathsf{dag}(\mathcal{R})$ and then checking whether π is covered by some path τ in $\mathsf{dag}^*(\mathcal{T})$, we compute a succinct representation of

all instances of the coverability relation. More precisely, we aim at representing the set P of all pairs (π, Π), where π is any path in $\mathsf{dag}(\mathcal{R})$ and Π is the set of all and only the paths in $\mathsf{dag}^*(\mathcal{T})$ that cover π. Note that this set P can be constructed inductively starting from the paths of length 1 and considering, at each step, the possible prolongations by single components. Because the set P might have size exponential in the number of components of $\mathcal{R}$, we need to succinctly represent it by abstracting all paths in it with their shallowest components. Formally, we replace any pair $(\pi, \{\tau_1, \ldots, \tau_\ell\})$ in P, where $\pi = \mathsf{C}_1 \ldots \mathsf{C}_k$ and $\tau_i = \mathsf{D}_{i,1} \ldots \mathsf{D}_{i,k}$ for all $1 \leq i \leq \ell$, with the corresponding pair $(\mathsf{C}_k, \mathsf{F})$, where $\mathsf{F} = \{\mathsf{D}_{1,k}, \ldots, \mathsf{D}_{\ell,k}\}$. This abstraction is correct because paths ending in the same component behave similarly with respect to the coverability properties of their prolongations. Algorithm 5.2 below implements such an idea.

Algorithm 5.2: AllPathsCoverability($\mathcal{R}$, $\mathcal{T}$)

```
let dag(R) = (V, E)
let dag*(T) = (V', E')
P ← ∅
for all C ∈ V
  do { F ← ∅
       for all D ∈ V'
         do { if CheckContainment(R, T, C, D)
              then F ← F ∪ {D} }
       P ← P ∪ {(C, F)} }
for k ← 2 to |V|
  do { for all (C, F) ∈ P and (C, C') ∈ E
       do { F' ← ∅
            for all D ∈ F and (D, D') ∈ E'
              do { if CheckContainment(R, T, C', D')
                   then F' ← F' ∪ {D'} }
            P ← P ∪ {(C', F')} }
return (∀ (C, F) ∈ P. F ≠ ∅)
```

The proof that the algorithm is correct, namely, that it terminates successfully iff every path π in $\mathsf{dag}(\mathcal{R})$ is covered by some path in $\mathsf{dag}^*(\mathcal{T})$, relies on the following invariant: at each iteration of the loop on k, the set P contains all pairs (C, F) such that (i) there exists a path $\pi \in \mathsf{dag}(\mathcal{R})$ that ends in C and has length at most k and (ii) the set F consists of all and only the shallowest components of the paths in $\mathsf{dag}^*(\mathcal{T})$ that cover π. 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 4.4). Finally, since the set P has size at most $|\mathcal{R}| \times 2^{|\mathcal{T}|}$ and $\mathcal{T}$ is fixed, we conclude that the algorithm runs in polynomial time with respect to the size of $\mathcal{R}$. □

5.2. The bounded repair problem in the streaming setting

DFA vs DFA. The characterization of Theorem 4.3 shows that the problem of deciding the existence of a streaming repair strategy with uniformly bounded

(aggregate or edit) cost for two languages specified by DFA $\mathcal{R}$ and $\mathcal{T}$ amounts at solving a reachability game over a suitable directed acyclic graph $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. In particular, we observe that $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ can be computed from $\mathcal{R}$ and $\mathcal{T}$ in polynomial time, and 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 directed acyclic graphs is PTIME-complete [7, 8]. This shows that the bounded repair problem for DFA in the streaming setting is PTIME-complete:

Corollary 5.6. *The bounded repair problem in the streaming setting, where the restriction and target languages are specified by DFA, is PTIME-complete.*

NFA vs NFA. Of course, the problem becomes more difficult when the languages are specified by NFA. In this case we are not able to provide tight complexity bounds, and we only claim that the complexity of the bounded repair problem for NFA in the streaming setting is between PSPACE and EXPTIME. The lower bound will follow from Corollary 5.10 below and the upper bound from the standard subset construction on NFA:

Corollary 5.7. *The bounded repair problem in the streaming setting, where the restriction and target languages are specified by NFA, is in EXPTIME and it is PSPACE-hard.*

DFA vs NFA. We analyse here the complexity of the bounded repair problem in the streaming setting where one of the two input automata is a DFA and the other is an NFA. In these cases, we can improve the upper bounds from EXPTIME to PSPACE:

Theorem 5.8. *The bounded repair problem in the streaming setting, where the restriction language is specified by a DFA and the target language is specified by an NFA, is PSPACE-complete.*
The bounded repair problem in the streaming setting, where the restriction language is specified by an NFA and the target language is specified by a DFA, is in PSPACE.

Proof. For both claims we make use of the characterization given by Theorem 4.3. We first deal with the case where the restriction language is given by a DFA $\mathcal{R}$ and the target language is given by an NFA $\mathcal{T}$. As a preliminary remark, observe that, in this case, PSPACE-hardness will follow again from Corollary 5.10 (below). As for the PSPACE upper bound, we denote by $\mathsf{det}(\mathcal{T})$ the DFA obtained from $\mathcal{T}$ by applying the standard subset construction, and we recall that, by Theorem 4.3, there is a streaming repair strategy for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ $(= \mathscr{L}(\mathsf{det}(\mathcal{T})))$ of uniformly bounded aggregate/edit cost iff Eve wins the reachability game over the arena $\mathcal{A}_{\mathcal{R},\mathsf{det}(\mathcal{T})}$. The crucial observation is that the longest collection of moves of Adam in the arena $\mathcal{A}_{\mathcal{R},\mathsf{det}(\mathcal{T})}$ is linear in the size of $\mathsf{dag}(\mathcal{R})$. This implies that the length of any play is at most $|\mathsf{dag}(\mathcal{R})|$ and hence we can simulate the reachability game over $\mathcal{A}_{\mathcal{R},\mathsf{det}(\mathcal{T})}$ by an

alternating polynomial-time procedure [7]. 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 D of $\mathsf{det}(\mathcal{T})$ (note that a SCC of $\mathsf{det}(\mathcal{T})$ can be specified by a single state of $\mathsf{det}(\mathcal{T})$ or, equivalently, by a set of states of $\mathcal{T}$). At each round of the reachability game, we need to check a language containment $\mathscr{L}(\mathcal{R}|\mathsf{C}) \subseteq \mathscr{L}(\mathsf{det}(\mathcal{T})|\mathsf{D})$: this can be done using the characterization given in Lemma 4.5 and a polynomial-space subroutine based on symbolic reachability analysis. What we have described is an alternating polynomial-time procedure that simulates the reachability game over $\mathcal{A}_{\mathcal{R},\mathsf{det}(\mathcal{T})}$ using a polynomial-space subroutine for language containment. Overall, this shows that, when the restriction language is given by a DFA, the bounded repair problem in the streaming setting is in PSPACE.

We now 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, we have to simulate the reachability game over the arena $\mathcal{A}_{\mathsf{det}(\mathcal{R}),\mathcal{T}}$ that results from the main characterization result. However, we cannot obtain a polynomial bound to the length of the plays since $\mathsf{dag}(\mathcal{R})$ has potentially exponential height. The idea here is that it is possible to modify the definition of the arena $\mathcal{A}_{\mathsf{det}(\mathcal{R}),\mathcal{T}}$ (and thus the resulting reachability game) by allowing Adam to move down the graph $\mathsf{dag}(\mathsf{det}(\mathcal{R}))$ using shortcuts, namely, by allowing Adam to move from any SCC of $\mathsf{det}(\mathcal{R})$ 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. On the other hand, if Adam wins in the modified arena, then he can also win in the original arena via longer plays. We now argue that, if Adam wins the modified reachability game, then he can do so with a polynomial number of moves. Indeed, a winning strategy of Adam consists in pushing Eve towards a sink node; this however can be done in at most n rounds, where n is the height of the DAG of SCCs of $\mathcal{T}$, by properly choosing shortcut moves. 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 repair problem for $\mathscr{L}(\mathcal{R})$ and $\mathscr{L}(\mathcal{T})$ in the streaming setting can be solved by an alternating polynomial-time procedure similar to the one described above. This shows that, when the target language is given by a DFA, the bounded repair problem in the streaming setting is in PSPACE. □

5.3. The bounded repair problem in the unrestricted case

We now consider the unrestricted case of the bounded repair problem, namely, the case where the restriction language is assumed to be Σ^* and the target language T is specified by a finite state automaton.

The following result adapts the characterization theorems presented in Section 4 to give a necessary and sufficient condition for bounded repairability in the unrestricted case. This result, which can be viewed as a special case of both Theorem 4.1 and Theorem 4.3, also shows that there is no difference between the non-streaming and the streaming settings when the restriction language is universal.

Corollary 5.9. *Given an alphabet Σ and an NFA $\mathcal{T}$, the following conditions are equivalent:*

1. $\mathsf{cost}(\Sigma^*, \mathscr{L}(\mathcal{T})) < \infty$,
2. $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|\mathsf{D})$ *for some component* D *of* $\mathcal{T}$,
3. $\mathsf{cost}^{\mathsf{aggr}}_{0-\mathsf{lookahead}}(\Sigma^*, \mathscr{L}(\mathcal{T})) \leq 2|\mathcal{T}|$.

Using the above characterization, one can easily devise a non-deterministic logarithmic-space algorithm that solves the bounded repair problem for DFA in the unrestricted (streaming or non-streaming) setting. Indeed, if the target automaton $\mathcal{T}$ is a DFA and D is a component of $\mathcal{T}$, then we have $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|\mathsf{D})$ iff for every symbol $\mathtt{a} \in \Sigma$ and every state $\mathtt{r} \in \mathsf{D}$, $\mathcal{T}$ contains a transition of the form $(\mathtt{r}, \mathtt{a}, \mathtt{r}')$, with $\mathtt{r}' \in \mathsf{D}$. 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 $\mathscr{L}(\mathcal{A}) \neq \varnothing$ iff Σ^* is repairable into $\mathscr{L}(\mathcal{A}')$ with uniformly bounded cost, where $\mathcal{A}'$ is a DFA that recognizes the language $\Sigma^* \, \mathscr{L}(\mathcal{A})$ and that can be constructed from $\mathcal{A}$ in logarithmic space.

In a similar way, one can show that the bounded repair problem for NFA in the unrestricted case is PSPACE-complete. This follows from Corollary 5.9 and from suitable reductions to/from the universality problem for NFA. Indeed, checking whether a target NFA $\mathcal{T}$ has a SCC D such that $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|\mathsf{D})$ 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 $(\Sigma \cup \{\#\})^*$ is repairable into $(\mathscr{L}(\mathcal{A}) \cdot \{\#\})^*$ with uniformly bounded cost and $\# \notin \Sigma$. Notice that a finite automaton $\mathcal{A}'$ recognizing the language $(\mathscr{L}(\mathcal{A}) \cdot \{\#\})^*$ can be computed in linear time from $\mathcal{A}$.

We thus conclude the following:

Corollary 5.10. *The bounded repair problem in the unrestricted case, where the target language is specified by a DFA (resp., NFA) is NLOGSPACE-complete (resp., PSPACE-complete).*

Another consequence of Corollary 5.9 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 $\mathtt{a} \in \Delta$ and every state $\mathtt{r}$ of $\mathcal{T}$, $\mathcal{T}$ contains a transition from $\mathtt{r}$ labelled by $\mathtt{a}$ (here we no more assume that $\mathcal{T}$ is trimmed). Consider a restriction alphabet $\Sigma \subseteq \Delta$ and suppose that Σ^* is *not* repairable into T with uniformly bounded cost. Let us consider a SCC D of $\mathcal{T}$ that is reachable from the initial state and terminal, namely, with no outgoing edges. We know that D does not contain any final state (otherwise, $\mathsf{cost}(\Sigma^*, \mathscr{L}(\mathcal{T})) < \infty$ would follow from Corollary 5.9). In this case, however, the same component D in the *complement* DFA $\mathcal{T}^{\mathsf{C}}$ would contain final states and hence Σ^* would be repairable into $\mathscr{L}(\mathcal{T}^{\mathsf{C}})$ with uniformly bounded cost. This shows that:

Corollary 5.11. *Given an alphabet* Σ *and a regular language* $T \subseteq \Delta^*$*, with* $\Sigma \subseteq \Delta$*, then either* $\mathsf{cost}(\Sigma^*, T) < \infty$ *or* $\mathsf{cost}(\Sigma^*, \Delta^* \setminus T) < \infty$*.*

5.4. The threshold problem in the non-streaming setting

We now consider the problem of calculating the exact repair cost in the non-streaming setting. If the restriction and target languages are specified by DFA, then we know from Theorem 5.2 that we can decide in coNP whether the worst-case repair cost is finite or infinite. Furthermore, Theorem 4.1 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 $\theta \in \mathbb{N}$, since then we can try every number θ below the polynomial bound. Perhaps surprising, this problem is harder than the bounded repair problem, although still within polynomial space:

Theorem 5.12. *The problem of determining whether* $\mathsf{cost}(R, T) \leq \theta$*, given two languages* R *and* T *specified by DFA and given a positive number* θ*, is PSPACE-complete. The same holds when* R *and* T *are specified by NFA.*

Proof. We first give the upper bound, assuming that R and T are recognized by some NFA $\mathcal{R}$ and $\mathcal{T}$. First, we recall that, by Theorem 4.1, we have either $\mathsf{cost}(R, T) \leq (1+|\mathsf{dag}(\mathcal{R})|)\cdot|\mathcal{T}|$ or $\mathsf{cost}(R, T) = \infty$. Thus, without loss of generality, we can assume that the threshold $\theta \in \mathbb{N}$ is represented in unary notation. We can then construct in polynomial time an NFA $\mathcal{R}_\theta$ that recognizes the language

$$R_\theta \stackrel{\text{def}}{=} \{v \in \Delta^* \ : \ \exists\, u \in R.\ \mathsf{dist}(u, v) \leq \theta\}.$$

The automaton $\mathcal{R}_\theta$ is defined by a suitable 'juxtaposition' of $\theta+1$ disjoint copies of $\mathcal{R}$:

- the states of $\mathcal{R}_\theta$ are the pairs (q, i), where q is a state of $\mathcal{R}$ and $0 \leq i \leq \theta$;
- the transitions of $\mathcal{R}_\theta$ are the triples $\big((q, i), b, (q', j)\big)$ that satisfy one of the following conditions
  1. $i = j$, $b \in \Sigma$, and (q, b, q') is a transition of $\mathcal{R}$,
  2. $i < j$, $b \in \Delta$, and $q = q'$,
  3. $i < j$, $b \in \Delta$, and there is a transition (q, a, q') of $\mathcal{R}$, with $a \in \Sigma$,
  4. $i < j$, $b \in \Sigma$, and there are two transitions (q, a, q'') and (q'', a', q') of $\mathcal{R}$, with $a, a' \in \Sigma$ and $b \in \{a, a'\}$;
- the initial (resp., final) states of $\mathcal{R}_\theta$ are the pairs (q, i), where q is an initial (resp., final) state of $\mathcal{R}$ and $0 \leq i \leq \theta$.

Note that the automaton $\mathcal{R}_\theta$ can be constructed in polynomial time from $\mathcal{R}$ and θ when θ is represented in unary. Moreover, $\mathcal{R}_\theta$ accepts all and only the words that are at distance at most θ from some words in R, and hence $\mathscr{L}(\mathcal{R}_\theta) = R_\theta$. Finally, it is easy to see that

$$\mathsf{cost}(R, T) \leq \theta \qquad \text{iff} \qquad R_\theta \subseteq T.$$

This reduces the threshold problem $\mathsf{cost}(R, T) \leq \theta$ to a containment problem between languages recognized by NFA, which is known to be in PSPACE [6].

We now discuss the PSPACE lower bound, which holds even for languages specified 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 $t_\bot$ and $t_\top$ for the bottom and top rows. A *tiling of height* N (for an instance $I = (n, S, H, V, t_\bot, t_\top)$) is a mapping g from the pairs $(i, j) \in [1, N] \times [1, n]$ to the tiles in S such that $g(1, j) = t_\bot$ and $g(N, j) = t_\top$ for all $1 \leq j \leq n$. We say that the tiling g *satisfies the constraints* of I if

1. $\big(g(i, j), g(i, j+1)\big) \in H$ for all $1 \leq i \leq N$ and all $1 \leq j < n$,
2. $\big(g(i, j), g(i+1, j)\big) \in V$ for all $1 \leq i < N$ and all $1 \leq j \leq n$.

The corridor tiling problem is the problem of deciding, given a tiling instance $I = (n, S, H, V, t_\bot, t_\top)$, whether there is a tiling g of some height $N \geq 1$ that satisfies the constraints in I. This problem is known to be PSPACE-complete [9].

Hereafter, we fix an instance $I = (n, S, H, V, t_\bot, t_\top)$ of the corridor tiling problem and a threshold $\theta \geq 1$ (note that for $\theta = 0$ the threshold problem becomes a containment problem between DFA, which is clearly solvable in polynomial time). Moreover, we assume that the threshold θ is represented in unary notation (clearly this does not make the threshold problem more difficult). Below, we reduce the corridor tiling problem for the instance I to a threshold problem for two DFA $\mathcal{R}$ and $\mathcal{T}$.

We let the restriction alphabet Σ consist of pairs of the form $\langle t, j \rangle$, where t is a tile from S and j is a number in $\{1, \ldots, n\}$. The restriction language R contains "$(\theta+1)$-redundant" encodings of tilings, namely, words of the form

$$\langle g(1,1), 1\rangle^{\theta+1} \ \ldots\ \langle g(1,n), n\rangle^{\theta+1} \ \ldots\ldots\ \langle g(N,1), 1\rangle^{\theta+1} \ \ldots\ \langle g(N,n), n\rangle^{\theta+1}$$

where N is a positive natural number and $g : [1, N] \times [1, n] \to S$ is a tiling of height N that satisfies the horizontal constraints and the constraints on the bottom and top rows (but possibly not the vertical constraints). We claim that the above language is recognized by a DFA $\mathcal{R}$ of size polynomial in I and in θ: 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 $\theta + 1$ and n (which clearly can be done with a number of states linear in θ and n).

We now turn to the definition of the target language. Its alphabet Δ contains all symbols of the restriction alphabet Σ plus marked symbols $\underline{\langle t, j \rangle}$, with $t \in S$ and $1 \leq j \leq n$. The marked symbols $\underline{\langle t, j \rangle}$ are used to highlight a violation of the vertical constraints in such a way that it becomes easy for an automaton to certify it. Intuitively, a violation of a vertical constraint is certified on a word $v \in \Delta^*$ when v contains a marked symbol of the form $\underline{\langle t, j \rangle}$ at some position x

and a symbol $\langle t', j\rangle$, with $(t, t') \notin V$, at position $x + (\theta + 1) \cdot n$ (this would imply that the tile t' is adjacent to t along the vertical axis). More precisely, the target language is defined as follows:

$$T =^{\text{def}} \bigcup_{\substack{(t,t')\notin V \\ 1\le j\le n}} \Sigma^* \ \underline{\langle t, j\rangle} \ \underbrace{\langle t, j\rangle^{\theta} \ \Sigma^{(\theta+1)\cdot(n-1)} \ \langle t', j\rangle}_{(\theta+1)\cdot n \text{ positions}} \ \langle t', j\rangle^{\theta} \ \Sigma^*.$$

It is easy to construct a DFA $\mathcal{T}$ of size polynomial in $|I|$ and θ that recognizes the language T.

We now argue that there is a repair strategy for R into T with cost uniformly bounded by θ iff every tiling $g : [1, N] \times [1, n] \to S$ violates the constraints in I. On the one hand, suppose that every tiling $g : [1, N] \times [1, n] \to S$ violates the constraints in I. Let us consider a word $u \in R$ of the form

$$\langle g(1,1), 1\rangle^{\theta+1} \ \ldots \ \langle g(1,n), n\rangle^{\theta+1} \ \ldots\ldots \ \langle g(N,1), 1\rangle^{\theta+1} \ \ldots \ \langle g(N,n), n\rangle^{\theta+1}$$

where g is a tiling of $[1, N] \times [1, n] \to S$. From the previous assumptions, we know that g violates the vertical constraints in I. This implies that there are $1 \le i < N$ and $1 \le j \le n$ such that $\big(g(i, j), g(i+1, j)\big) \notin V$, and hence there are two positions x $(= j+(i-1)\cdot(\theta+1)\cdot n)$ and $x+(\theta+1)\cdot n$ $(= j+(i-1)\cdot(\theta+1)\cdot n+(\theta+1)\cdot n)$ in u that contain the symbols $\langle g(i, j), j\rangle$ and $\langle g(i + 1, j), j\rangle$, respectively. Marking the first of these two occurrences (i.e., replacing the symbol $\langle g(i, j), j\rangle$ at position x with the symbol $\underline{\langle g(i, j), j\rangle}$) will bring us into the target language T. This shows that there is a repair strategy for R and T with worst-case cost at most 1 $(\le \theta)$.

On the other hand, suppose that there is a tiling $g : [1, N] \times [1, n] \to S$ that satisfies the constraints in I. Let u be the corresponding word:

$$\langle g(1,1), 1\rangle^{\theta+1} \ \ldots \ \langle g(1,n), n\rangle^{\theta+1} \ \ldots\ldots \ \langle g(N,1), 1\rangle^{\theta+1} \ \ldots \ \langle g(N,n), n\rangle^{\theta+1}.$$

Clearly, u belongs to the restriction language R. We claim that no matter how we repair u by at most θ edits, the resulting sequence will not belong to the target language T. Consider a word v produced by such an edit. If there are no occurrences of marked symbols in v, then v cannot belong to T, and similarly if there are more than one occurrence of a marked symbol. Suppose that v contains exactly one occurrence of a marked symbol, say $v[x] = \underline{\langle t, j\rangle}$, and let $v[x + (\theta + 1) \cdot n] = \langle t', j'\rangle$. We distinguish between the following cases:

1. $v[x + i] \ne \langle t, j\rangle$ for some $1 \le i \le \theta$,
2. $j' \ne j$ or $v[x + (\theta + 1) \cdot n + i] \ne \langle t', j'\rangle$ for some $1 \le i \le \theta$,
3. $j' = j$ and $v[x + i] = \langle t, j\rangle$ and $v[x + (\theta + 1) \cdot n + i] = \langle t', j\rangle$ for all $1 \le i \le \theta$.

In the first two cases, v cannot belong to T. In the third case, since v was obtained from u by applying at most θ edit operations, we know that u contains a sub-string of the form

$$\langle t, j\rangle^{\theta+1} \ \Sigma^{(\theta+1)\cdot(n-1)} \ \langle t', j\rangle^{\theta+1}.$$

Since u is an encoding of a valid tiling g, we have that $(t, t') \in V$. This shows again that v cannot belong to the target language T. □

5.5. The threshold problem in the streaming setting

For the streaming setting, if we consider 0-lookahead repair strategies with aggregate cost, the threshold problem becomes solvable in polynomial time (hence it is as difficult as the bounded repair problem). Indeed, one can easily reduce this threshold problem to a reachability game over a suitable arena that is constructed in polynomial time from the restriction and target DFA $\mathcal{R}$ and $\mathcal{T}$ and from the threshold θ. It follows from this reduction that one can efficiently compute in polynomial time a streaming repair strategy for $\mathcal{R}$ and $\mathcal{T}$ whose aggregate cost does not exceed the threshold θ.

Theorem 5.13. *The problem of determining whether* $\mathsf{cost}^{\mathsf{aggr}}_{0-\mathsf{lookahead}}(R,T) \le \theta$*, given two languages* R *and* T *specified by DFA and given a numbers* θ*, is in PTIME.*
Moreover, if $\mathsf{cost}^{\mathsf{aggr}}_{0-\mathsf{lookahead}}(R,T) \le \theta$*, then one can compute in polynomial time a streaming* 0*-lookahead repair strategy for* R *and* T *that has aggregate cost at most* θ*.*

Proof. Let $\mathcal{R} = (\Sigma, Q, \delta, q_0, F)$ and $\mathcal{T} = (\Delta, Q', \delta', r_0, F')$ be two DFA, let R and T be the recognized languages, and let θ be the threshold for the aggregate cost. As a preliminary remark, we observe that, without loss of generality, we can assume that θ is represented in unary: indeed, we know from Theorem 4.3 and Corollary 4.6 that either there is a streaming repair strategy for R and T with aggregate cost uniformly bounded by $(1+|\mathsf{dag}(\mathcal{R})|)\cdot|\mathcal{T}|$ (i.e., a polynomial in the size of $\mathcal{R}$ and $\mathcal{T}$), or all streaming repair strategies for R and T have unbounded aggregate cost.

We define a reachability game over an arena $\mathcal{A}^{\theta}_{\mathcal{R},\mathcal{T}}$ that characterizes the threshold problem for $\mathcal{R}$ and $\mathcal{T}$ in the streaming setting. The nodes of the arena are the pairs (q,r,c) and (q,r,c,a), with $q \in Q$, $r \in R$, $c \in \{0,\dots,\theta\}$, and $a \in \Sigma$. 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 $\mathcal{A}^{\theta}_{\mathcal{R},\mathcal{T}}$ has an edge $(q,r,c) \to (q',r,c,a)$ if $\delta(q,a) = q'$, and it has an edge $(q,r,c,a) \to (q,r',c')$ if r' is reachable from r in $\mathcal{T}$ and $c' = c + \min\{\mathsf{dist}(a,v) : v \in \mathscr{L}(\mathcal{T}_{r,r'})\}$ (provided that $c' \le \theta$). Adam plays first, starting from the node $(q_0, r_0, 0)$. The player who cannot move loses. Infinite plays, which are feasible in this type of game, are won by Eve.

Now, we show that Eve has a winning strategy in $\mathcal{A}^{\theta}_{\mathcal{R},\mathcal{T}}$ iff there is a streaming 0-lookahead repair strategy for R and T with aggregate cost at most θ. Easily, assume that Eve has a strategy f for winning the reachability game over $\mathcal{A}^{\theta}_{\mathcal{R},\mathcal{T}}$. Without loss of generality, we can assume that the strategy f is positional. It is then easy to construct, using Eve's positional strategy f, a transducer $\mathcal{S}$ with at most $|Q|\cdot|Q'|\cdot(\theta+1)$ states that repairs R into T with aggregate cost $\mathsf{cost}^{\mathsf{aggr}}_{\mathcal{S}}(R,T) \le \theta$. For the other direction, suppose that there is a transducer $\mathcal{S} = (\Sigma, \Delta, Q'', \delta'', s_0, \Omega)$ that repairs every word from $\mathcal{R}$ into $\mathcal{T}$ with aggregate cost less than or equal to θ. It is straightforward to define a winning strategy for Eve using the transducer $\mathcal{S}$. Indeed, we only need to maintain the

current state s of $\mathcal{S}$ and move from each node (q, r, c, a) to a successor node (q, r', c') such that $r' = \delta(r, \nu)$, $c' = c + \mathsf{dist}(a, \nu)$, and $\delta''(s, a) = (\nu, s')$.

To conclude the proof we need show that, assuming $\mathsf{cost}^{\mathsf{aggr}}_{0-\mathsf{lookahead}}(R, T) \leq \theta$, one can compute in polynomial time a streaming 0-lookahead repair strategy for R and T that has aggregate cost at most θ. However, this follows immediately from the fact that (i) given the arena $\mathcal{A}^{\theta}_{\mathcal{R},\mathcal{T}}$ whose reachability game is won by Eve, one can construct in polynomial time a winning positional strategy f for Eve, and (ii) using Eve's winning strategy f, one can construct in polynomial time a repair strategy $\mathcal{S}$ for R and T such that $\mathsf{cost}^{\mathsf{aggr}}_{\mathcal{S}}(R, T) \leq \theta$ (for this we use again the above arguments). □

Note that in the above result we deal with the model of *aggregate* cost, which is very different the model of edit cost (the main characterization result, however, shows that one is uniformly bounded iff the other is). We do not know if finding the exact *edit* cost for two languages specified by DFA is even tractable. Similarly, we do not know whether the threshold problem for streaming repair strategies with k-*lookahead* is tractable, where the parameter k is represented in binary.

6. Connections to distance automata and games

Both the non-streaming and streaming repair problems correspond to special cases of prior problems studied in automata and games. The non-streaming bounded repair problem corresponds to the *limitedness problem* for distance automata, while the streaming variant corresponds to *energy games*. We explain the correspondences in detail now. In each case, however, we find that the results for the more general framework do not give tight complexity bounds.

Non-streaming repairs and distance automata. Intuitively, a *distance automaton* is a transducer $\mathcal{D}$ that receives as input a finite word u and outputs a corresponding cost $\mathcal{D}(u)$ in $\mathbb{N}^{\infty} = \mathbb{N} \cup \{\infty\}$. Formally, a distance automaton is a transducer of the form $\mathcal{D} = (\Sigma, Q, E, I, F)$, where Σ is the input alphabet, Q is a finite set of states, $E \subseteq Q \times \Sigma \times \mathbb{N}^{\infty} \times Q$ is the transition relation, and $I, F : Q \to \mathbb{N}^{\infty}$ are the initial and final cost functions. The cost $\mathcal{D}(u)$ on input $u = a_1 \dots a_n \in \Sigma^*$ is obtained by taking the minimum among the costs of the runs of $\mathcal{D}$ on u, where the cost of each run $q_0 \xrightarrow{a_1/c_1} q_1 \xrightarrow{a_2/c_2} \dots \xrightarrow{a_n/c_n} q_n$ is defined as $I(q_0) + \sum_{i=1}^{n} c_i + F(q_n)$ – in particular, every transition of the run has a cost, as well as the beginning and the end of the run. We let $\mathcal{D}(u) = \infty$ 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 there exists a finite upper bound to the cost $\mathcal{D}(u)$ computed by a given distance automaton $\mathcal{D}$ for all words $u \in \Sigma^*$ such that $\mathcal{D}(u) < \infty$. This problem was shown decidable by Hashigushi [10]; later in [11] it was shown to be PSPACE-complete. Distance automata have been related to edit-distance problems in several prior works – see Section 7 for further discussion of the connections. Here we only provide a simple reduction of the bounded repair problem to the limitedness problem.

Given two NFA $\mathcal{R} = (\Sigma, Q, E, I, F)$ and $\mathcal{T} = (\Delta, Q', E', I', F')$, one can construct a distance automaton $\mathcal{D}$ that computes the optimal cost of repairing any word from $\mathscr{L}(\mathcal{R})$ into a word from $\mathscr{L}(\mathcal{T})$. First of all, one associates 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 $\mathcal{T}$ can move from p to q consuming v (if q is not reachable from p, then one simply lets $M(a)[p, q] = \infty$). One then defines the distance automaton $\mathcal{D}$ as the quadruple $(\Sigma, Q \times Q', E^M, I^M, F^M)$, where E^M is the set of all transitions of the form $\big((p, p'), a, c, (q, q')\big)$, with $a \in \Sigma$, $(p, a, q) \in E$, and $c = M(a)[p', q']$. The initial cost function I^M of $\mathcal{D}$ is defined by letting $I^M(p')$ be the length of the minimum word that can be parsed by $\mathcal{T}$ when moving from an initial state to the state p' (if no such word exist, then $I^M(p') = \infty$). Similarly, $F^M(p)$ is defined to be the length of the minimum word that can be parsed by $\mathcal{T}$ when moving from the state p to a final state (or ∞ if no such word exists). It is easy to see that the cost function computed by $\mathcal{D}$ maps a word $u \in \mathscr{L}(\mathcal{R})$ to the cost of an optimal 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 $\mathcal{T}$. This reduces the bounded repair problem for NFA in the non-streaming setting to the limitedness problem for distance automata. 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 coNP (cf. Theorem 5.2). Roughly speaking, the reason why the bounded repair problem for DFA is easier than the limitedness problem for distance automata is that the distance automata emerging from bounded repair problems have a more restricted structure (specifically, they 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 bounded repairability, Theorem 4.1, with the PSPACE upper bound given in [11].

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 graphs with quantitative objectives. An interesting family of such games is that of *energy games* studied in [12], which are played on finite weighted arenas. The game is played between an energy player, who wants to keep the running sum of the weights (i.e. the energy) always positive, 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.

It is well known 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 [13], but the exact complexity is still unknown. Furthermore, this problem can be solved in time polynomial in the size of the arena and the largest weight in

absolute value. As a matter of fact, the latter complexity result implies that energy games 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 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. More 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 pair of states reached by the restriction and target automata after parsing the unedited and edited words, respectively; 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) whenever (p, a, q) is a valid transition of $\mathcal{R}$. Similarly, there is an edge of weight $-c$ going from a Target Player Node (q, p', a) to a Restriction Player Node (q, q') whenever there is a word v at distance c from a (i.e. $\mathsf{dist}(a, v) = c$) such that $\mathcal{T}$ can move from p' to q' consuming v. Clearly, the energy player wins the game using some initial credit of energy if and only if the cost of repairing $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ is uniformly bounded. Note that this reduction provides a PTIME upper bound to the complexity of the bounded streaming repair problem for DFA, given 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.

Our characterization result (see Theorem 4.3) gives analogous (tight) complexity bounds for bounded repairability languages recognized by DFA and, moreover, it proves 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, which cannot naturally be captured in the setting of energy games. It is also worth mentioning that the repair strategy that arises from our characterization result can be seen as a special case of the notion of good-for-energy strategy, which is introduced in [13] to solve energy parity games.

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 lookahead 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 finally allows us to easily isolated special cases of lower complexity that are not easily seen from the embedding into energy games.

	fixed	**DFA**	**NFA**
universal	LOGSPACE	NLOGSPACE	PSPACE
fixed	Const	PTIME	PSPACE
DFA	PTIME	coNP	PSPACE
NFA	PTIME	coNP	PSPACE

Table 1: Bounded repair problem in the non-streaming setting

	fixed	**DFA**	**NFA**
universal	LOGSPACE	NLOGSPACE	PSPACE
fixed	Const	PTIME	PSPACE
DFA	PTIME	PTIME	PSPACE
NFA	PTIME-PSPACE	PTIME-PSPACE	PSPACE-EXPTIME

Table 2: Bounded repair problem in the streaming setting

7. Conclusions and Related Work

In this work we have investigated the problem of repairing documents between different specifications. In our basic setting, a document is represented by a string and a specification by a language. Specifically, we gave a characterization of those pairs of regular languages such that one can repair any string in the first language to a string in the second, using a uniformly bounded number of edits and repair processors that either read the entire string offline and then edit, or have the form of sequential real-time transducers. We then used the characterizations to provide complexity bounds for the considered bounded repair problems. These complexity bounds are summarized in Table 1 and Table 2 – in the non-streaming setting all bounds are tight (indicated by a single class), while in the streaming setting we have several gaps (indicated by cells with lower and upper bounds). 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 also know the complexity of computing the optimal stream repair processor.

Related Work. 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, giving a polynomial time algorithm. Several authors have extended the definition to deal with distances between languages. Mohri [14] defines a distance function between two languages, 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 [15] focuses on the minimum distance between distinct strings within the same language, giving tractable algorithms for computing it. Our notion of "distance" is quite distinct from this, since it is asymmetric in the two lan-

guages, focusing on the maximum of the distance of a string in one language to the other language. As a matter of fact, this notion of distance between languages can be seen as a special case of a more general concept, known as Hausdorff distance. This latter notion has been studied also in [16], but within a more general setting; in particular, with respect to a generalization of the boundedness property called "almost reflexivity".

Grahne and Thomo [17] consider a related problem of "approximate containment" of regular expressions. Expressions are evaluated with respect to an edge-labelled 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. languages) – see, for example, [18]. However, in those cases the lifting is done by looking at the syntactic structure of the schema description, rather than at the instance level (distance between documents in each schema).

An extended abstract of this work appeared in [19].

References

[1] M. Arenas, L. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), 1999, pp. 68–79.

[2] F. Afrati, P. Kolaitis, Repair checking in inconsistent databases: algorithms and complexity, in: Proceedings of the 12th International Conference on Database Theory (ICDT), 2009, pp. 31–41.

[3] R. Wagner, Order-$\mathfrak{n}$ correction for regular languages, CACM 17 (5) (1974) 265–268.

[4] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley Longman Publishing Co., Inc., 1979.

[5] M. Schützenberger, Sur une variante des fonctions sequentielles, Theoretical Computer Science 4 (1) (1977) 47–57.

[6] L. Stockmeyer, A. Meyer, Word problems requiring exponential time: preliminary report, in: Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC), 1973, pp. 1–9.

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

[8] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: a guide to current research, Vol. 2500 of Lecture Notes in Computer Science, Springer, 2002.

[9] P. Van Emde Boas, The convenience of tilings, in: Complexity, Logic and Recursion Theory, Vol. 187, 1997, pp. 331–363.

[10] K. Hashiguchi, Improved limitedness theorems on finite automata with distance functions, Theoretical Computer Science 72 (1) (1990) 27–38.

[11] H. Leung, V. Podolskiy, The limitedness problem on distance automata: Hashiguchi's method revisited, Theoretical Computer Science 310 (2004) 147–158.

[12] A. Chakrabarti, L. de Alfaro, T. Henzinger, M. Stoelinga, Resource interfaces, in: Proceedings of the 3rd International Conference on Embedded Software (EMSOFT), 2003, pp. 117–133.

[13] K. Chatterjee, L. Doyen, Energy parity games, Theoretical Computer Science 458 (2012) 49–60.

[14] M. Mohri, Edit-distance of weighted automata: general definitions and algorithms, International Journal of Foundations of Computer Science 14 (6) (2003) 957–982.

[15] S. Konstantinidis, Computing the edit distance of a regular language, Information and Computation 205 (9) (2007) 1307–1316.

[16] C. Choffrut, G. Pighizzini, Distances between languages and reflexivity of relations, Theoretical Computer Science 286 (1) (2002) 117–138.

[17] G. Grahne, A. Thomo, Query answering and containment for regular path queries under distortions, in: Proceedings of the 3rd International Symposium on Foundations of Information and Knowledge Systems (FOIKS), 2004, pp. 98–115.

[18] H. Do, E. Rahm, COMA - a system for flexible combination of schema matching approaches, in: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), 2002, pp. 610–621.

[19] M. Benedikt, G. Puppis, C. Riveros, Regular repair of specifications, in: Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS), 2011, pp. 335–344.