Regular Repair of Specifications

Michael Benedikt
Oxford University
michael.benedikt@comlab.ox.ac.uk

Gabriele Puppis
Oxford University
gabriele.puppis@comlab.ox.ac.uk

Cristian Riveros
Oxford University
cristian.riveros@comlab.ox.ac.uk

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

I. Introduction

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

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

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

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

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 function that given a word w runs a dynamic-programming algorithm to compute the edit distance to the target (e.g. the algorithm from [3] in the case of NFAs). 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 setting, and also give procedures for computing the optimal processor in the streaming setting.

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

In summary our contributions are:

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

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

II. Basic notation and terminology

Let Σ be a finite alphabet and Σ^* be the set of finite words over Σ. We denote the empty word by ε and the length of a word $w \in \Sigma^*$ by $|w|$.

Automata. Non-deterministic finite state automata (shortly, *NFA*) will be represented by tuples of the form $\mathcal{A} = (\Sigma, Q, E, I, F)$, where Σ is a finite alphabet, Q is a finite set of states, $E \subseteq Q \times \Sigma \times Q$ is a transition relation, and $I, F \subseteq Q$ are sets of initial and final states. By $\mathscr{L}(\mathcal{A})$ we denote the language recognized by $\mathcal{A}$. If $\mathcal{A}$ is a deterministic finite state automaton (*DFA*), then we usually denote the unique initial state by q_0 and turn its transition relation E into a partial function δ from $Q \times \Sigma^*$ to Q defined by $\delta(q, \varepsilon) = q$ and $\delta(q, a \cdot u) = \delta(q', u)$ iff $(q, a, q') \in E$. For technical reasons, we assume that all states of an NFA are reachable from some initial state and from all states a final state is reachable. It is worth noticing that, since the decision problems we are going to deal with are at least NLOGSPACE-hard and since any given automaton can be pruned using NLOGSPACE, this assumption will have no impact on our complexity results.

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

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

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

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

III. Problem setting

Given two finite alphabets Σ and Δ, we denote by $\mathrm{dist}(u, v)$ the *Levenshtein distance* (henceforth, edit distance) between two words $u \in \Sigma^*$ and $v \in \Delta^*$, which is defined as the length of a shortest sequence s of edit operations (e.g., deleting, modifying, and inserting a single character) that transforms u into v [4]. A *processor* is simply a function from Σ^* to Δ^*. For a processor f, we refer to $\mathrm{dist}(u, f(u))$ as *the cost of f on the word u*. Given a language $R \subseteq \Sigma^*$, we define the *worst-case cost* of f over R as the supremum of the cost of f over all words in R. If the cost is unbounded, then we say that the worst-case cost is ω.

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

The *bounded repair problem* is to decide, given languages R and T, whether $\operatorname{dist}(R,T)$ is finite, that is, whether there is a repair strategy f of R into T and a natural number $n \in \mathbb{N}$ such that $\operatorname{dist}(u, f(u)) \le n$ for all $u \in R$. Similarly, the *threshold problem* is to compute the exact value of $\operatorname{dist}(R,T)$. Clearly, the languages R and T must be finitely represented, for instance, in terms of machines or logical formulas. In this paper, we study the complexity of the bounded repair problem for input languages represented by means of the following formalisms: (i) deterministic finite state automata (DFA), (ii) non-deterministic finite state automata (NFA), and (iii) LTL formulas with only future modal operators.

Streaming vs non-streaming. In its most general formulation, a repair strategy could be any function mapping words to edit words. However, we know from [3] that there is a dynamic programming algorithm that, given a word u and a target language T represented by a DFA, computes in polynomial time an optimal edit sequence s such that $s(u) \in T$. In particular, this shows that optimal repair strategies can be described by functions of fairly low complexity. Sometimes it is desirable to have repair strategies that are in even more limited classes. Perhaps the ideal case is when a strategy is realizable with a bounded memory one-pass algorithm, that is, using a (letter-to-word sequential) transducer. Recall that a letter-to-word transducer defines a word-to-word function (i.e., a processor); if this function is a repair strategy, we refer to the transducer as a *streaming repair strategy*. The idea is that any input word u from a restriction language should be repaired in an online way. Similarly, we can talk about a k-lookahead streaming repair strategy.

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

Example 1. *Consider* $R = (a+b)c^*(a^+ + b^+)$ *and* $T = ac^*a^+ + bc^*b^+$. *In the non-streaming case, one can get from* R *to* T *by only editing the initial letter and, thus,* $\operatorname{dist}(R,T)$ *is equal to* 1. *In contrast, a streaming repair strategy must decide whether to leave or change the initial letter, and then it could be forced to repair an unbounded sequence of* a *or* b *after the sequence of* c.

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

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

The first cost considers the distortions performed in producing the input from the output – it is equivalent to considering the transducer as producing edits rather than strings and counting the number of edits produced. The second cost is global and it considers only the output and not its production. Clearly, the last cost never exceeds the aggregate cost.

It is important to notice that these two models of cost can be very different in general. Consider a transducer $\mathcal{S}$ on the input alphabet $\Sigma = \{a, b\}$ that swaps a's and b's. On the string $u_n = (ab)^n$, the aggregate cost is $2n$ since $\mathcal{S}$ changes each letter, but the edit cost between u and $\mathcal{S}(u)$ is only 2. Nevertheless, it will turn out that for the bounded repair problem it does not matter which model of cost we choose (see Theorem 3).

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

A. Repair Problems, Automata, and Games

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

Non-streaming repairs and distance automata. Intuitively, a *distance automaton* is a transducer $\mathcal{D}$ that receives as input a finite word w and outputs a corresponding cost $\mathcal{D}(w)$ in $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$. 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 $\mathcal{D}(w)$ on input $w = a_1 \dots a_n \in \Sigma^*$ is obtained by taking the minimum among the costs of the runs of $\mathcal{D}$ on w, where the cost of a 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 $I(q_0) + \sum_{i=1}^{n} c_i + F(q_n)$. We let $\mathcal{D}(w) = \infty$ if $\mathcal{D}$ admits no successful run on w.

The main problem that has been studied for distance automata is the *limitedness problem* which consists of deciding whether the cost function computed by a given distance automaton $\mathcal{D}$ is uniformly bounded on all words $w \in \Sigma^*$ with cost $\mathcal{D}(w) \neq \infty$. This problem was shown decidable by Hashigushi [5] and later in [6] was shown to be PSPACE-complete. Distance automata have been related to edit-distance problems in several prior works – see Section IX for further discussion of the connections. Here we note only a simple reduction of the bounded repair problem to limitedness. Given two NFA $\mathcal{R}$ and $\mathcal{T}$, one can construct a distance automaton $\mathcal{D}$ that computes the cost of repairing any word from $\mathscr{L}(\mathcal{R})$ into a word from $\mathscr{L}(\mathcal{T})$. Let $\mathcal{R} = (\Sigma, Q, E, I, F)$ and $\mathcal{T} = (\Delta, Q', E', I', F')$ be two NFA for the restriction and target languages. First of all, we associate with each symbol $a \in \Sigma$ a matrix $M(a)$ whose entries $M(a)[p,q]$ are indexed over the pairs of states p, q of $\mathcal{T}$ and give the minimum edit-distance between the symbol a and a word $v \in \Delta^*$ such that $\mathcal{T}$ can move from p to q consuming v. If q is not reachable from p, then we let $M(a)[p,q] = \infty$. We then define the distance automaton $\mathcal{D}$ as the quadruple $(\Sigma, Q \times Q', E^M, I^M, F^M)$, 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 $c = M(a)[p',q']$. Further, we define $I^M(p,p')$ as the length of the minimum word from a state in I' to p' if $p \in I$ and ∞ otherwise. Similarly, $F^M(p,p')$ is the length of the minimum word from p' to a state in F' if $p \in F$ and ∞ otherwise. It is easy to see that the cost function computed by $\mathcal{D}$ maps a word $u \in \mathscr{L}(\mathcal{R})$ (which is accepted by $\mathcal{D}$ too) to the cost of the best non-streaming repair of u into $\mathscr{L}(\mathcal{T})$. Moreover, the distance automaton $\mathcal{D}$ has size polynomial in the size of $\mathcal{R}$ and $\mathcal{T}$. Combining this reduction with the PSPACE upper bound for the limitedness problem, we see that the bounded repair problem for NFA is in PSPACE.

The same reduction technique can be applied to solve the bounded repair problem for DFA. In this case, however, the resulting complexity bound is not optimal: the bounded repair problem for DFA is in fact in coNP (cf. Corollary 4). Roughly speaking, the reason why the bounded repair problem for DFAs is easier than the limitedness problem for distance automata is that the distance automata emerging from DFA repair problems are deterministic on the 0-cost moves. In addition to not giving tight bounds, approaches via distance automata give less insight into the problems. We invite the reader, for example, to compare the PSPACE upper bound that we derive from our characterization of repairability, Theorem 2, with the PSPACE upper bound given in [6].

Streaming repairs and energy games. Just as non-streaming repair problems can be seen within the framework of distance automata, bounded repair problems in the streaming setting are special cases of games on graph with quantitative objectives. An interesting family of such games is that of *energy games* studied in [7], which are played on finite weighted arenas. The game is played between an energy player, who wants to mantain the 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 [8], 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. Formally, we have a node in the arena for each pair of states of the restriction DFA $\mathcal{R}$ and of the target DFA $\mathcal{T}$ – call this node a "Restriction Player Node". We also have a node for each combination of restriction state, target state, and letter played – call this a "Target Player Node". The former represents the states reached by the restriction and target automata after parsing the unedited and edited words, while the latter adds the last letter emitted by the restriction. There is an edge of weight 0 going from a Restriction Player Node (p,p') to any Target Player Node (q,p',a), where (p,a,q) is a transition of the restriction DFA $\mathcal{R}$. Similarly, there is an edge of weight $-c$ going from a Target Player Node (q,p',a) to a Restriction Player Node (q,q') provided that there is a word v at distance c from a (i.e., $\mathrm{dist}(a,v) = c$) such that $\mathcal{T}$ can move from p' to q' consuming v. It is important to observe 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 results (see Theorem 3) give analogous (tight) complexity bounds for languages recognized by DFA and moreover, prove that the bounded repair problem in the streaming setting is not sensitive to the models of aggregate/edit cost. They also provide tight bounds for special cases of the problem that cannot naturally be captured in the setting of energy games. Our repair strategy can be seen as a special case of the notion of good-for-energy strategy, which is introduced in [8] 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 also allows us to easily isolated special cases of lower complexity that are not easily seen from the embedding into energy games.

IV. Characterizations of bounded repairability

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

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

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

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

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

The interesting directions are from 2) to 3) and from 1) to 2). For the first implication, if the coverability condition is satisfied, then we repair a word $w \in \mathscr{L}(\mathcal{R})$ by choosing any path $\pi = C_1 \dots C_n$ in $\mathrm{dag}(\mathcal{R})$ taken by a run of w, and looking at a covering path in $\mathrm{dag}^*(\mathcal{T})$. We can consider $w = u_1 a_1 u_2 \dots a_{n-1} u_n$ such that $u_i \in \mathscr{L}(\mathcal{R}|C_i)$ and $a_j \in \Sigma$ for all $i \leq n$ and $j < n$. For a covering path $\pi' = C'_1 \dots C'_n$ of π this implies that $u_i \in \mathscr{L}(\mathcal{T}|C'_i)$. Therefore, at the boundary points a_i where w jumps from the SCC C_i to the next SCC C_{i+1} in $\mathcal{R}$, we can insert small words that push the computation from C'_i to C'_{i+1} in $\mathcal{T}$; because these are strongly connected components and there is a path from C'_i to C'_{i+1}, we can arrange a jump to any state in C'_{i+1}. Thus we can repair w by inserting a bounded number of small words and adding a small word at the end to reach a final state in $\mathcal{T}$.

The second implication is more complex, and is proven by contraposition. Assuming the negation of 2) we know that there is a path $\pi = C_1 \dots C_k$ of $\mathrm{dag}(\mathcal{R})$ that is not covered by paths in $\mathrm{dag}^*(\mathcal{T})$. For each SCC C_i of π we construct a word u_i that witnesses all non-containments of $\mathscr{L}(\mathcal{R}|C_i)$ in SCCs of $\mathcal{T}$. We then construct, for each n, a word w_n formed by concatenating n-fold iterations of each word u_i, that is, $w_n = u'_0 \ u_1^n \ u'_1 \ \dots \ u_k^n \ u'_k$ where the fixed words $u'_0, \dots, u'_k$ are arranged to make sure the resulting word is in $\mathscr{L}(\mathcal{R})$. Finally, we argue that w_n requires at least n edits to be repaired into a word in $\mathscr{L}(\mathcal{T})$.

The streaming case. We now modify Theorem 2 to give a characterization of the streaming repair problem, adding in a game setting. Fix two DFAs $\mathcal{R}$ and $\mathcal{T}$ recognizing the restriction and target languages. We associate with the DFA a *reachability game* between two players, Adam and Eve, on a suitable arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$, defined in terms of the SCCs of $\mathcal{R}$ and $\mathcal{T}$. The idea underlying this game is as follows: during Adam's construction of a path π in $\mathrm{dag}(\mathcal{R})$, Eve has to provide a construction of a corresponding path $f(\pi)$ in $\mathrm{dag}^*(\mathcal{T})$ that covers π; moreover, the resulting function f must satisfy the following condition: if $\pi \cdot C$ is an extension of the path π in $\mathrm{dag}(\mathcal{R})$ by a single SCC, then either $f(\pi \cdot C)$ coincides with $f(\pi)$ or it is an extension of $f(\pi)$ by a single SCC, namely, $f(\pi \cdot C)$ is of the form $f(\pi) \cdot D$.

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

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

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

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

Going from 2) to 3), if we have a strategy for Eve, we can get a streaming repair strategy by tracking the current SCC C of the input string and maintaining the invariant that the component of the current repaired string D is such that (C, D) is a position consistent with Eve's winning strategy. When a new letter comes in and changes the SCC in the restriction from C to C', we respond with a repair that moves from D to the response SCC D' that preserves the invariant.

For the direction from 1) to 2), we assume a k-lookahead repair strategy and derive a strategy for Eve; our strategy will maintain the invariant that the position (C, D) corresponds to some input string w and response w' consistent with the repair strategy. If, by way of contradiction, we have such a pair (C, D) corresponding to some string w, a successor SCC C' of C corresponding to some extension wu of w and (D, C') is a lossing position for Eve, then we can construct a single counterexample word for every candidate SCC. Given that for every successor SCC D' of D there is $v \in \mathscr{L}(\mathcal{R}|C') \setminus \mathscr{L}(\mathcal{T}|D')$, we can concatenate multiple copies of v together. If we make the number of copies large enough, such a string cannot be repaired by our transducer with a bounded number of edit operations, a contradiction.

V. Complexity results in the non-streaming case

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

A. The bounded repair problem

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

NFA. Theorem 2 gives a straightforward PSPACE algorithm that solves the bounded repair problem between two NFA $\mathcal{R}$ and $\mathcal{T}$ in this setting: the algorithm first guesses universally a path $\pi = C_1...C_n$ in $\mathrm{dag}(\mathcal{R})$, then it guesses existentially a path $\pi' = C'_1...C'_n$ of the same length in $\mathrm{dag}^*(\mathcal{T})$, and finally it checks the containment of the subautomaton $\mathcal{R}|C_i$ in the subautomaton $\mathcal{T}|C'_i$ for all indices $1 \leq i \leq n$. Together with the PSPACE lower bound for the problem proven later (see Corollary 19), we obtain:

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

DFA. The same characterization result can be used to solve the problem when the restriction language is represented by an NFA and the target language is represented by a DFA. In this case, we can take advantage of the determinism to show that the problem turns out to be coNP-complete. Intuitively, the complexity upper bound follows from the observation that containment of languages recognized by SCCs of DFA is decidable in PTIME even if the successful runs can start from arbitrary states inside the SCCs and that the above mentioned coverability problem for paths of SCCs is in coNP. In other words, we can guess a path in $\mathrm{dag}(\mathcal{R})$ and check in PTIME if this path is not covered in $\mathrm{dag}^*(\mathcal{T})$. The complexity lower bound follows from a reduction from the validity problem for propositional formulas in disjunctive normal form (i.e., the dual of the SAT problem): the idea is to encode in the restriction language all the possible valuations for the propositional variables and then restrict the target language to consist only of encodings of valuations that satisfy at least one clause of the formula. Notice that some redundancy is needed in the restriction to forbid the repair strategy from modifying the encoded valuations.

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

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

It is more difficult to show that the bounded repair problem is tractable when we fix the target automaton. Here, instead of guessing a path π in $\mathrm{dag}(\mathcal{R})$ and then checking whether π is covered by some path π' in $\mathrm{dag}^*(\mathcal{T})$, we directly compute all instances of the coverability relation. We then perform a top-down algorithm to compute which restriction components are covered.

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

LTL. We conclude the section by analyzing the complexity of the bounded repair problem where languages defined by LTL formulas are involved. We first consider the problem where both the restriction language R and the target language T are defined by some LTL formulas ϕ and ψ. It is not difficult to see that this problem is in coNEXPTIME. Indeed, one can use standard automaton-based techniques to construct, in exponential time, two *DFA* $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$ that recognize the *reversals* $\overleftarrow{R}$ and $\overleftarrow{T}$ of the languages R and T. Since, in the non-streaming setting, the cost of repairing R into T is the same as the cost of repairing $\overleftarrow{R}$ into $\overleftarrow{T}$, one can exploit Theorem 5 to solve the bounded repair problem on the DFA $\overleftarrow{\mathcal{R}}$ and $\overleftarrow{\mathcal{T}}$ in coNEXPTIME. For the complexity lower bound, one can reduce the problem of deciding the non-existence of a tiling of an exponential square grid, which is known to be coNEXPTIME-complete [9], to the problem of deciding the existence of a repair strategy of uniformly bounded cost between two regular languages defined by suitable LTL formulas. The idea of such a reduction is to let the formula for the restriction language encode all candidate tilings and the formula for the target language check that none of them is correct.

Theorem 7. *The bounded repair problem in the non-streaming case, where the restriction and target languages are represented by LTL formulas, is coNEXPTIME-complete.*

The bounded repair problem becomes easier when it involves repairs of languages recognized by NFA into languages defined by LTL formulas. The idea is to convert the formula into a symbolic automata (represented using propositional formulas), and then apply the characterization theorem, looking for paths in the NFA that are not covered by paths in the symbolically-represented target language. Because the required containment checks can be done in PSPACE on the symbolic representations, we get:

Theorem 8. *The bounded repair problem in the non-streaming case, where the restriction language R is represented by an NFA and the target language T is represented by an LTL formula, is in PSPACE.*

Similarly, the bounded repair problem remains in PSPACE when the restriction is specified by an LTL formula ϕ and the target is recognized by an NFA $\mathcal{T}$. In this case, one still uses Theorem 2 and a symbolic DFA $\overleftarrow{\mathcal{R}}$ recognizing the reversal of the language defined by ϕ. However, instead of universally guessing an entire path π in $\mathrm{dag}(\overleftarrow{\mathcal{R}})$ one guesses the leaf of a counterexample path, and verifies that it is not covered by moving down from the root to the leaf.

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

B. The threshold problem

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

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

Theorem 10. *The problem of determining, given k and two languages R and T recognized by DFA, whether $\mathrm{dist}(R,T)$ is above k, is PSPACE-complete. The same holds when R and T are given as an NFA.*

The upper bound is shown by reachability analysis in a product automata representing all states reachable via at most k edits. The lower bound uses a reduction from tiling a polynomial width corridor. Roughly speaking, our restriction language will represent codes of potential tilings, with each tile repeated k times. Our target language will check that the word still codes a tiling k-redundantly, and will also check for markings on tiles that indicate that a violation of horizontal or vertical constraints lies within a k-neighborhood of the marked tile. If there is no accepting run, then every potential tiling can be marked with a constraint violation. Conversely, if the restriction is repairable, then it can be shown that marking must correctly indicate violations on every candidate tiling.

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

Theorem 11. *The problem of determining, given k and two languages R and T defined by LTL formulas, whether $\mathrm{dist}(R,T)$ is above k, is EXPSPACE-complete.*

The lower bound is proven using a variation of the tiling technique in the previous theorem.

VI. Complexity results in the streaming case

A. The bounded repair problem

DFA. Let us consider two DFA $\mathcal{R}$ and $\mathcal{T}$. The characterization of Theorem 3 shows that the problem of deciding the existence of a streaming repair strategy of $\mathscr{L}(\mathcal{R})$ into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost amounts at solving a reachability game over a suitable (acyclic) arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$. In particular, we observe that the arena $\mathcal{A}_{\mathcal{R},\mathcal{T}}$ can be computed from $\mathcal{R}$ and $\mathcal{T}$ in polynomial time 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 acyclic graphs is PTIME-complete [10]. This shows that the bounded repair problem for DFA in the streaming case is PTIME-complete:

Corollary 12. *The bounded repair problem in the streaming case, where the restriction and target languages are represented by DFA, is PTIME-complete.*

It is worth noticing that the complexity of the bounded repair problem for DFA in the streaming setting is lower than the analogous problem in the non-streaming setting (indeed Theorem 5 shows that the latter problem is coNP-complete). This will be in contrast with the complexity results for languages defined by LTL formulas, where the streaming setting becomes more difficult than the non-streaming setting (compare Theorem 5 and Theorem 15).

NFA. When both restriction and target are NFA we are not able to provide tight complexity bounds, thus we only claim that the complexity of the bounded repair problem for NFA is between PSPACE and EXPTIME. The lower bound follows from Corollary 19 and the upper bound from the standard subset construction on NFA:

Corollary 13. *The bounded repair problem in the streaming case, where the restriction and target languages are represented by NFA, is in EXPTIME and it is PSPACE-hard.*

In the case where the restriction is a DFA $\mathcal{R}$ and the target is an NFA $\mathcal{T}$, we obtain a tight PSPACE bound. PSPACE-hardness follows again from Corollary 19. As for the PSPACE upper bound, we observe that the longest collection of moves of Adam in the arena $\mathcal{A}_{\mathcal{R},\mathrm{det}(\mathcal{T})}$, where $\mathrm{det}(\mathcal{T})$ denotes the DFA obtained from $\mathcal{T}$ by applying the standard subset construction, is linear in the size of $\mathrm{dag}(\mathcal{R})$. By representing each SCC of $\mathrm{det}(\mathcal{T})$ using a set of states from $\mathcal{T}$, one obtains an alternating polynomial-time procedure that simulates the reachability game over $\mathcal{A}_{\mathcal{R},\mathrm{det}(\mathcal{T})}$.

In the symmetric case, where the restriction is an NFA and the target is a DFA, one could prove an EXPTIME upper bound on the bounded repair problem via reduction to *energy games with imperfect information* (studied by Degorre et. al. in [11]). However, we can improve this upper bound to PSPACE by simulating a reachability game over the arena $\mathcal{A}_{\mathrm{det}(\mathcal{R}),\mathcal{T}}$. In this case the crucial observation is that it is safe to modify the arena $\mathcal{A}_{\det(\mathcal{R}),\mathcal{T}}$ by allowing Adam to move down the DAG of $\det(\mathcal{R})$ with shortcuts, namely, from a SCC of $\mathcal{R}$ to any descendant of it (rather than simply a successor of it). 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. Moreover, if Adam wins the modified reachability game, then he can do so in at most $|\mathrm{dag}^*(\mathcal{T})|$ rounds by properly choosing shortcut moves that push Eve towards a sink node. This shows that the problem is in PSPACE (we do not know whether it is also PSPACE-hard).

Theorem 14. *The bounded repair problem in the streaming case, where the restriction language is a DFA and the target language is an NFA, is PSPACE-complete. The bounded repair problem in the streaming case, where the restriction language is an NFA and the target language is a DFA, is in PSPACE.*

LTL. We now turn to the complexity of the bounded repair problem in the streaming case, where both restriction and target languages are represented by LTL formulas. By following standard constructions in automata theory, one can translate any pair of LTL formulas ϕ and ψ into DFA $\mathcal{R}$ and $\mathcal{T}$ that have size doubly exponential in the size of the formulas ϕ and ψ and that recognize the same languages defined by ϕ and ψ. This gives a straightforward 2EXPTIME upper bound to the complexity of the bounded repair problem. As for the complexity lower bound, we can reduce the problem of deciding the winner of tiling game over an exponential square grid – this problem is known to be EXPSPACE-complete [9] – to the problem of deciding the existence of a streaming repair strategy of uniformly bounded cost between the languages defined by suitable LTL formulas (the idea of such a reduction is similar to the coNEXPTIME-hardness proof of Theorem 7):

Theorem 15. *The bounded repair problem in the streaming case, where the restriction and target languages are given by LTL formulas, is in 2EXPTIME and is EXPSPACE-hard.*

B. The threshold problem and constructing streaming repairs

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

Theorem 16. *The problem of determining, given k and two languages R and T recognized by DFA, whether one can repair R into T with a streaming repair strategy with aggregate cost at most k, is in PTIME.*

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

Corollary 17. *Let R and T be the restriction and target languages specified by DFA. If R is streaming repairable into T with aggregate cost at most k, then an optimal streaming repair strategy of R into T with aggregate cost at most k can be computed in PTIME.*

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

VII. Special cases: unrestricted repair problems

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

The following result adapts the characterization theorems given in Section IV to give a necessary and sufficient condition for the unrestricted case. This result, which can be viewed as a special case of both Theorem 2 and Theorem 3, also shows that there is no difference between the non-streaming and the streaming settings when the restriction language is universal.

Corollary 18. *Given an alphabet Σ and an NFA $\mathcal{T}$, the following conditions are equivalent*

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

Using the above characterization, one can easily devise an NLOGSPACE algorithm that solves the bounded repair problem for DFA in the unrestricted (streaming or non-streaming) case. Indeed, if the target automaton $\mathcal{T}$ is a DFA and C is a component of $\mathcal{T}$, then we have $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|C)$ iff for every symbol $a \in \Sigma$ and every state q in C, $\mathcal{T}$ contains a transition of the form (q, a, q'), with $q' \in C$. Checking this property amounts to performing a standard NLOGSPACE reachability analysis over $\mathcal{T}$. Conversely, NLOGSPACE-hardness follows from the fact that the emptiness problem for DFA is reducible to the bounded repair problem: given a DFA $\mathcal{A}$ over an alphabet Σ, we have that $\mathscr{L}(\mathcal{A}) \neq \emptyset$ iff Σ^* is repairable into $\mathscr{L}(\mathcal{A}')$ with uniformly bounded cost, where $\mathcal{A}'$ is a DFA that 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 18 and from suitable reductions from the universality problem for NFA. Indeed, checking whether a target NFA $\mathcal{T}$ has a SCC C such that $\Sigma^* \subseteq \mathscr{L}(\mathcal{T}|C)$ is equivalent to the problem of deciding whether Σ^* is repairable into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost, and it is clearly reducible to the universality problem for NFA. As for the PSPACE-hardness, we observe that a given NFA $\mathcal{A}$ recognizes the universal language Σ^* iff $(\Sigma \uplus \{\#\})^*$ is repairable into $(\mathscr{L}(\mathcal{A}) \cdot \{\#\})^*$ with uniformly bounded cost. Notice that a finite automaton for the language $(\mathscr{L}(\mathcal{A}) \cdot \{\#\})^*$ can be computed in linear time.

We thus conclude the following:

Corollary 19. *The bounded repair problem in the unrestricted case, where the target languages are represented by DFA (resp., NFA) is NLOGSPACE-complete (resp., PSPACE-complete).*

Another consequence of Corollary 18 is the following. Suppose that a target language T is recognized by a DFA $\mathcal{T}$ that is *complete* over the target alphabet Δ, namely, for every symbol $a \in \Delta$ and every state p of $\mathcal{T}$, $\mathcal{T}$ contains a transition from p labeled by a. Let us consider a restriction alphabet Σ contained in Δ and suppose that Σ^* is *not* repairable into T with uniformly bounded cost. Let us consider a SCC C of $\mathcal{T}$ that is reachable from the initial state and terminal, namely, with no outgoing edges. We know that C does not contain any final state (otherwise, C would be a final SCC and hence, by Corollary 18, Σ^* would be repairable into $\mathscr{L}(\mathcal{T})$ with uniformly bounded cost). In this case, however, the same component C in the complement DFA $\mathcal{T}^{\complement}$ would be final and hence Σ^* would be repairable into $\mathscr{L}(\mathcal{T}^{\complement})$ $(= \Delta^* \setminus T)$ with uniformly bounded cost. This shows that:

Corollary 20. *Given an alphabet Σ and a regular language $T \subseteq \Delta^*$, with $\Sigma \subseteq \Delta$, one of the following two cases (possibly both) holds:*

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

We now turn to the complexity of the bounded repair problem in the unrestricted case, but where the target languages are represented by LTL formulas. We claim that problem is PSPACE-hard for LTL formulas. This complexity lower bounds follows from arguments similar to the automaton-based setting, namely, from a reduction of the satisfiability problem for LTL formulas, which is known to be PSPACE-hard [12]. As for the complexity upper bound, we claim that the problem for LTL formulas is in PSPACE and, thus, PSPACE-complete. Indeed, given an LTL formula ψ defining a target language T, one can compute in polynomial time a symbolic representation of a DFA $\overleftarrow{\mathcal{T}}$ that recognizes the *reversal* $\overleftarrow{T}$ of T. Moreover, one can perform standard reachability analysis on the symbolic representation of $\mathcal{T}$ in polynomial space. Finally, we observe that Σ^* is repairable into T with uniformly bounded cost iff Σ^* is repairable into $\overleftarrow{T}$ with uniformly bounded cost. This shows that the bounded repair problem in the unrestricted case for LTL formulas is in PSPACE.

Corollary 21. *The bounded repair problem in the unrestricted case, where the target languages are represented by LTL formulas, is PSPACE-complete.*

VIII. Towards infinite words

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

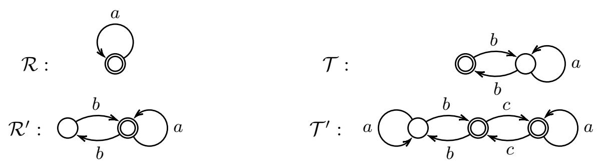

Figure 1: Some non-deterministic Büchi automata.

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

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

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

1) all paths in Π' have length precisely $k+1$,
2) $\mathscr{L}(\mathcal{R}|\pi(i)) \subseteq \bigcap_{\pi' \in \Pi'} \mathscr{L}(\mathcal{T}|\pi'(i))$ for all indices $i < k$,
3) $\mathscr{L}^{\omega}(\mathcal{R}|^{\omega}\pi(k)) \subseteq \bigcap_{\pi' \in \Pi'} \mathscr{L}(\mathcal{T}|\pi'(k)) \cdot \bigcup_{\pi' \in \Pi'} \mathscr{L}^{\omega}(\mathcal{T}|^{\omega}\pi'(k{+}1))$.

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

Theorem 22. *Given two NBA $\mathcal{R}$ and $\mathcal{T}$, the following conditions are equivalent*

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

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

Table I: Complexity of bounded non-streaming repair

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

Table II: Complexity of bounded streaming repair

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

IX. Related Work and Conclusions

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

We have focused on the case of finite words, but infinite words raise many new issues. In the case of infinite words in a streaming setting, one can look for strategies that allow finitely many edits per word, without a uniform bound, and likewise look for strategies with "continuous" (but not uniformly-bounded) lookahead. This last issue has been investigated for purely qualitative games by Holtmann et. al. [13].

Related work on edit distance of languages. The problem of finding the minimal distance of a string to a regular language was first considered by Wagner in [3], who showed that the problem could be solved by adapting the dynamic programming approach to edit distance, giving a polynomial time algorithm. Several authors have extended the definition to deal with distances between languages. Mohri [14] defines a distance function between two sets of strings, and more generally between string distributions: in the case of languages, this is the minimum distance between two strings in the two respective languages, which is appropriate for many applications. Konstantinidis [15] focuses on the minimum distance between distinct strings within the same language, giving tractable algorithms for computing it. Our notion of "cost" is quite distinct from this, since it is asymmetric in the two languages, focusing on the maximum of the distance of a string in one language to the other language.

Grahne and Thomo [16] consider a related problem of "approximate containment" of regular expressions. Expressions are evaluated with respect to an edge-labeled graph and are given a numerical semantics by a "distortion" – a generalization of the notion of edit distance. Approximate containment of T_1 and T_2 means, roughly speaking, that for every input graph R and every word w generated by R, the distance to target T_1 is bounded by the distance to T_2. Grahne and Thomo also study "k-containment" (distance to T_1 is at most k more than T_2) and "approximate-containment" (k-containment for some k), relying primarily on a reduction to the limitedness problem for distance automata. Their problem differs in several fundamental respects from ours: they are interested in bounding the difference over all words, not just the worst case; in addition they quantify over all restrictions (databases, in their terminology).

An entire line of research in XML data management has dealt with comparisons and matching algorithms between schema languages; many of these lift edit distance between trees to the level of schemas (i.e. languages) – see, for example, [17]. However 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).

Acknowledgments. We thank the anonymous referees and Slawek Staworko for many helpful comments. The authors were supported by EPSRC (UK) grant EP/G004021/1.

References

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

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

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

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

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

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

[7] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and M. Stoelinga, "Resource interfaces," in *EMSOFT*, 2003, pp. 117–133.

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

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

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

[11] A. Degorre, L. Doyen, R. Gentilini, J. Raskin, and S. Toruńczyk, "Energy and mean payoff games with imperfect information," in *CSL*, 2010, pp. 260–274.

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

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

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

[15] S. Konstantinidis, "Computing the edit distance of a regular language," *Inf. and Comp.*, vol. 205, no. 9, pp. 1307–1316, 2007.

[16] G. Grahne and A. Thomo, "Query answering and containment for regular path queries under distortions," in *FOIKS*, 2004, pp. 98–115.

[17] H. Do and E. Rahm, "COMA - a aystem for flexible combination of schema matching approaches," in *VLDB*, 2002, pp. 610–621.