How to Travel between Languages[⋆]

Krishnendu Chatterjee[1] and Siddhesh Chaubal[2] and Sasha Rubin[1,3]

[1] IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria
`Krishnendu.Chatterjee@ist.ac.at`, `sasha.rubin@ist.ac.at`
[2] IIT Bombay, Powai, Mumbai 400076, India `spchaubal@gmail.com`
[3] TU Wien, Institut für Informationssysteme 184/4, Favoritenstraße 911, 1040, Vienna, Austria

Abstract. We consider how to edit strings from a source language so that the edited strings belong to a target language, where the languages are given as deterministic finite automata. Non-streaming (or offline) transducers perform edits given the whole source string. We show that the class of deterministic one-pass transducers with registers along with increment and min operation suffices for computing optimal edit distance, whereas the same class of transducers without the min operation is not sufficient. Streaming (or online) transducers perform edits as the letters of the source string are received. We present a polynomial time algorithm for the *partial-repair problem* that given a bound α asks for the construction of a deterministic streaming transducer (if one exists) that ensures that the 'maximum fraction' η of the strings of the source language are edited, within cost α, to the target language.

Keywords: Edit-distance, Markov decision process, register automaton

1 Introduction

One of the classic problems in language theory concerns optimally editing an input string so that it belongs to a given target regular language [3].

Definition 1 (Edit-distance). *An* edit operation *applied to a string u either deletes a single character, inserts a single character, or changes a single character. The* edit-distance *between $u, v \in \Sigma^*$, denoted $\mathrm{ED}(u, v)$, is defined as the length of a shortest sequence of edit operations that applied to u yields v. For language T, we define $\mathrm{ED}(u, T) := \inf_{v \in T} \mathrm{ED}(u, v)$.*

In [2] the problem was generalized: given source language R and target language T, how to edit strings from R so that the edited strings belong to T.

[⋆] The research was supported by Austrian Science Fund (FWF) Grant No P 23499-N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), and Microsoft faculty fellows award. Thanks to Gabriele Puppis for suggesting the problem of identifying a deterministic transducer to compute the optimal cost, and to Martin Chmelik for his comments on the introduction.

There are two broad categories of transducers $\mathfrak{Tr}$ that edit strings depending on how they process input. The *non-streaming (or offline)* transducers reads the complete source string $u \in R$ and then output the repaired string $\mathfrak{Tr}(u) \in T$. The *streaming (or online)* transducers perform the edits as the letters of input u are received. For a class of transducers, the main interest is to compare $u \mapsto \textsc{ed}(u, \mathfrak{Tr}(u))$ — ie. the cost of editing using $\mathfrak{Tr}$ from the class — to $u \mapsto \textsc{ed}(u, T)$, the cost of optimal editing.

Previous results for non-streaming transducers: Wagner [3] gives a polynomial time algorithm for computing the optimal edit-distance, namely $u \mapsto \textsc{ed}(u, T)$, given a DFA for T. In [2] it was shown (Section $III.A$) that certain (non-deterministic) distance automata can compute this optimal cost.

Our contributions for non-streaming transducers: We consider the problem of finding a natural class of *deterministic* transducer for computing the optimal edit-distance $u \mapsto \textsc{ed}(u, T)$. We observe that Wagner's algorithm can be reformulated using cost-register automata of [1]. Specifically, one can use the deterministic one-pass transducers with registers that allow parallel updates using the increment and arbitrary-arity minimum operations. We prove that natural restrictions of this model, notably by disallowing use of the minimum operator, do not suffice to compute the optimal edit-distance (Theorem 5). This uses some pumping-like arguments.

Previous results for streaming transducers: In [2] it was also shown that whether there is a streaming transducer (Definition 7) with finite streaming-cost (Definition 8) that repairs strings from R to strings in T is a PTIME-complete problem (the languages R, T are given as DFAs). Moreover, if the cost is finite, then a streaming transducer can be extracted from their proof [2][Theorem 3].

Our contributions for streaming transducers: We consider the problem of repairing strings from source to target language in the case that the streaming-cost is infinite or very large. In this case we fix an edit-distance bound α and ask what is the 'largest fraction' of strings $\eta \in [0, 1]$ that can be repaired within cost α by a streaming transducer. This is called the *partial-repair problem.* We show with an example (Example 12) that although the streaming-cost is infinite, a large fraction (formalised in Definition 11) of the strings from the source language may be edited with small edit distance to belong to the target language. Our main contribution (Theorem 14) is a polynomial time algorithm solving the partial-repair problem, and, if one exists, a construction of a corresponding deterministic streaming transducer. We do this by building and solving a Markov decision process whose value is the largest such η.

2 Non-streaming Transducers

Following Wagner [3], there is a dynamic programming algorithm that given a string $u \in \Sigma^*$ and a DFA for target language $T \subset \Sigma^*$ can compute, in PTIME, the integer $\textsc{cost}(u, T)$ (and a string $t \in T$ with the property that $\textsc{cost}(u, T) = \textsc{ed}(u, t)$). In [2][Section III] it is mentioned that this gives a transducer of fairly low complexity. We formalise this intuition and observe that there is a *deterministic* transducer model that implements Wagner's approach and computes $u \mapsto \text{ED}(u, T)$. In fact, the transducers are certain cost register automata (introduced in [1]), which we call *repair-transducers.* A repair-transducer is a DFA with a fixed number of register names K. Write int_k for the value, a natural number, of register $k \in K$. Initially each register contains 0. At each step the DFA reads the next letter from the input string, updates its local state, and makes a parallel update to the registers. The allowed updates may mention the register names, the constant 0, addition by a constant, and the minimum of any number of terms.[1] Once the input string has been read the machine outputs the contents of some of its registers. Thus a repair-transducer realises a function $\Sigma^* \to \mathbb{N}^k$. Formally, an update is a term generated by the grammar:

```
inc-min ::= 0 | int_k  (for k∈K) | inc-min + c  (for c∈ ℕ)
inc-min ::= min(inc-min, inc-min,...,inc-min)
```

If $\nu : K \to \mathbb{N}$ and τ is an inc-min term, write $[[\tau]]_\nu$ for the evaluation of term τ under assignment ν.[2]

Definition 2. *A* repair-transducer $\mathfrak{Tr}$ *is a DFA* (Σ, Q, q_0, δ) *without final states augmented by a finite set* K *of register names, a register update function* $\mu : Q \times \Sigma \times K \to IMT$*, and a final register function* $f : Q \to K$*, where* IMT *is the set of* $inc - min$ *terms. A* configuration *is an element of* $Q \times \mathbb{N}^K$*. The initial configuration is* (q_0, ν_0) *where* $\nu_0(k) = 0$ *for all* $k \in K$*. The* run *on input* $u = u_1 \cdots u_n \in \Sigma^*$ *is the sequence of configurations* $(q_0, \nu_0) \cdots (q_n, \nu_n)$ *such that* $q_{i+1} = \delta(q_i, u_i)$ *(for* $0 \leq i < n$*) and for each* $k \in K$, $\nu_{i+1}(k) := [[\mu(q_i, u_i, k))]]_{\nu_i}$*. The transducer outputs* $\mathfrak{Tr}(u) := \nu_n(f(q_n))$*.*

Proposition 3. *For regular language* T *there is a repair-transducer* $\mathfrak{Tr}$ *computing* $u \mapsto \text{COST}(u, T)$*. Moreover, given a DFA for* T *one can build* $\mathfrak{Tr}$ *in PTIME.*

For the proof simply note that the dynamic-programming identities in Wagner's algorithm only use the $+c$ and min operations. Clearly if we disallow use of $+c$ operation then the transducer can't accumulate values and so can't compute $u \mapsto \text{ED}(u, T)$. What if we disallow the min operation?

Proposition 4. *There is a language* T *such that every repair-transducer for* T *requires the use of* min *in its update rules.*

Proof. Let $T = 0^* + 1^*$. Suppose there were a repair transducer $\mathfrak{Tr}$ for T with state set Q, register set K, but no use of the min operation. Let $\delta : Q \times \Sigma^* \to Q$ be the transition function (extended to strings) and $M : Q \times \Sigma^* \times \mathbb{N}^K \to \mathbb{N}^K$ the evaluated update function extended to strings. That is: the run of $\mathfrak{Tr}$ starting from (q, ν) on input $u \in \{0, 1\}^*$ ends in $(\delta(q, u), M(q, u, \nu))$. Note that $\mathfrak{Tr}(w)$ is the minimum of the number of 0s in w and the number of 1s in w.

[1] This is similar to the inc-min grammar of [1]. However there they only allow a binary min operation.

[2] Namely, $[[0]]_\nu := 0$, $[[int_k]]_\nu := \nu(k)$, $[[\tau + c]] := [[\tau]] + c$, and $[[min(\tau_1, \cdots, \tau_n)]] := \min\{[[\tau_1]], \cdots, [[\tau_n]]\}$.

Fact 1. For every (q, u) there are functions $F_{q,u} : K \to K$ and $\#F_{q,u} : K \to \mathbb{N}$ such that for all ν, k,

$$M(q, u, \nu)(k) = \nu(F_{q,u}(k)) + \#F_{q,u}(k), \tag{1}$$

in words: the kth component of $M(q, u, \nu)$ is equal to the $F_{q,u}(k)$th component of ν plus integer $\#F_{q,u}(k)$. This can be proved by induction on $|u|$ and uses the fact that $\mathfrak{Tr}$ only has $+c$ updates.

Fact 2. For every $F_{q,u} : K \to K$ there exists $N, P \in \mathbb{N}$ such that for all $y \in \mathbb{N}$, $(F_{q,u})^N = (F_{q,u})^{N+Py}$. This follows from the fact that the set of functions $K \to K$ is a finite set closed under composition so apply the pigeonhole principle.

Fact 3. Fix u, q such that $\delta(q, u) = q$, and N, P from Fact 1 applied to $F_{q,u}$. For every $k \in K$ there exists $j \in K$ and $\alpha, \beta \in \mathbb{N}$ such that for all $\nu \in \mathbb{N}^K$ and $y \in \mathbb{N}$, $M(q, u^{N+Py}, \nu)(k) = \beta + y\alpha + \nu(j)$. For the proof use: $\beta := \#F_{q,u^N}(k)$, $j := F^N_{q,u}(k)$, and $\alpha := \#F_{q,u^P}(j)$.

We now apply some pumping arguments. Note that in the lemma if $\alpha = 0$ then pumping doesn't increase the value. In this case call the triple (u, q, k) *flat*. On the other hand if $\alpha > 0$ then pumping increases the value. In this case call the triple (u, q, k) *increasing*. We exploit this dichotomy.

By the pigeonhole principle there exists $q_0 \in Q$ and $c, d \in \mathbb{N}$ such that $\delta(\iota, 0^a) = q_0$, $\delta(q_0, 0^b) = q_0$. Thus $\delta(\iota, 0^{a+bx}) = q_0$ for all $x \in \mathbb{N}$. Similarly, there exists $q_1 \in Q$, $c, d \in \mathbb{N}$ such that $\delta(q_0, 1^c) = q_1$, $\delta(q_1, 1^d) = q_1$. Thus $\delta(q_1, 0^{c+dy}) = q_1$ for all $y \in \mathbb{N}$. Write $w_{x,y} := 0^{a+bx}1^{c+dy}$. Note that $\mathfrak{Tr}(w_{x,y}) = \min\{a + bx, c + dy\}$. Let $k := f(q_1)$ be the register whose value is output when given strings of the form $w_{x,y}$.

Suppose x is much bigger than y (technically: $a + bx > c + d(y + 1)$). Then after reading input $w_{x,y}$ register k has value $c + dy$. And after reading input $w_{x,y+1}$ register k has value $c + d(y + 1)$. This implies that the triple $(1^d, q_1, k)$ is increasing. On the other hand, suppose x is smaller than y. Then after reading input $w_{x,y}$ register k has value $a + bx$. And after reading input $w_{x,y+1}$ register k has value $a + bx$. This implies that the triple $(1^d, q_1, k)$ is flat. But a triple can not be both increasing and flat. □

We summarise the results of this section:

Theorem 5. *For every regular language $T \subset \Sigma^*$ there is a repair-transducer $\mathfrak{Tr}$ computing* $\textsc{cost} : \Sigma^* \to \mathbb{N}, u \mapsto \textsc{cost}(u, T)$. *Moreover, the models of transducer which disallow the $+c$ operators or the* min *operator cannot, in general, compute this cost.*

3 Streaming Transducers

Here is the generalisation of edit-distance to a source and target language:

Definition 6 (Repair-cost). *[2] Given two languages $R, T \subset \Sigma^*$ define the* repair-cost from R to T *as* $\textsc{cost}(R \rightsquigarrow T) := \sup_{u \in R} \inf_{v \in T} \textsc{ed}(u, v)$.

Note the asymmetry in the definition of repair cost: it expresses the worst-case cost of repairing all strings in R to strings in T. The cost may be finite $\textsc{cost}((ab)^* \rightsquigarrow (ba)^*) = 2$ — just delete the first and last letter of the input — or infinite $\textsc{cost}(a^* \rightsquigarrow (ba)^*) = \infty$ — since $\textsc{ed}(a^{2n}, (ba)^*) = n$.

Definition 7 (Streaming Transducer). *A* streaming transducer *is a device of the form* $\mathfrak{Tr} = (\Sigma, \Sigma_{out}, Q, \delta, q_0, \Omega)$*, where*

- Σ *is a finite* input alphabet*, and* Σ_{out} *is a finite* output alphabet*,*
- Q *is a finite set of* states *and* $q_0 \in Q$ *is an* initial state*,*
- δ *is a* transition function $Q \times \Sigma \to \Sigma_{out}^* \times Q$*, and*
- Ω *is a* final-output function $Q \to \Sigma_{out}^*$.

For every string $u = a_1 \dots a_n$ *in* Σ^**, there is a unique sequence of states* $q_0, q_1, \cdots, q_n$ *and strings* $v_1, \cdots, v_n$ *such that* $\delta(q_i, a_{i+1}) = (v_{i+1}, q_{i+1})$ *for all* $0 \le i < n$*. In this case define the output of* $\mathfrak{Tr}$ *on* u *to be the string* $\mathfrak{Tr}(u) = v_1 v_2 \dots v_n v_{n+1}$ *where* $v_{n+1} = \Omega(q_n)$*. We write* $q_0 \xrightarrow{a_1/v_1} q_1 \xrightarrow{a_2/v_2} \cdots \xrightarrow{a_n/v_n} q_n \xrightarrow{v_{n+1}}$.

Define the output of $\mathfrak{Tr}$ *on language* R *to be the set* $\mathfrak{Tr}(R) = \{\mathfrak{Tr}(u) \mid u \in R\}$.

Definition 8 (Streaming Cost[3]). *[2] For a streaming transducer* $\mathfrak{Tr}$ *and an input string* $u = a_1 \dots a_n \in \Sigma^*$*, if the run of* u *on* $\mathfrak{Tr}$ *is* $q_0 \xrightarrow{a_1/v_1} q_1 \xrightarrow{a_2/v_2} \cdots \xrightarrow{a_n/v_n} q_n \xrightarrow{v_{n+1}}$*, then the* streaming cost *of* $\mathfrak{Tr}$ *on* u *is defined as:*

$$\textsc{cost}_{\mathfrak{Tr}}(u) = |v_{n+1}| + \sum_{i=1}^{n} \textsc{ed}(a_i, v_i)$$

For language R *define* $\textsc{cost}_{\mathfrak{Tr}}(R) := \sup_{u \in R} \textsc{cost}_{\mathfrak{Tr}}(u)$.

If $\mathfrak{Tr}(R) \subset T$ *then say that* $\mathfrak{Tr}$ *is a* streaming transducer from R to T.

Note. $\textsc{ed}(u, \mathfrak{Tr}(u)) \le \textsc{cost}_{\mathfrak{Tr}}(u)$ and so $\textsc{cost}(R \rightsquigarrow \mathfrak{Tr}(R)) \le \textsc{cost}_{\mathfrak{Tr}}(R)$. The streaming-cost is an upper bound on the repair-cost. So if $\textsc{cost}(R \rightsquigarrow T) = \infty$ then there is no streaming-transducer from R to T with finite streaming-cost.

Example 9. [2] Let $\Sigma = \{a, b, c\}$, $R = (a+b)c^*(a^+ + b^+)$ and $T = ac^*a^+ + bc^*b^+$. Then for every $r \in R$ there is $t \in T$ such that $\textsc{ed}(r, t) \le 1$ (correct the first letter if required). However, for every streaming transducer $\mathfrak{Tr}$ from R to T, $\textsc{cost}_{\mathfrak{Tr}}(R) = \infty$. In other words, the cost of repairing all strings in R to strings in T is finite, but is not realisable by a streaming transducer with finite streaming-cost.

In the last example consider a streaming transducer $\mathfrak{Tr}$ that outputs an 'a' and then copies the rest of the input (ie. it sends $(a+b)c^n w$ to $ac^n w$). It correctly repairs strings of the form $(a+b)c^*a^+$ (incurring cost ≤ 1) and incorrectly strings of the form $(a+b)c^*b^+$; that is, it is a streaming transducer from $(a+b)c^*a^+$

[3] In [2] this is called the *aggregate cost* of $\mathfrak{Tr}$ on u.

to T. Then, informally, $\mathfrak{Tr}$ repairs with cost at most 1 half the strings of R to T; and formally $\mathfrak{Tr}$ is a $(\frac{1}{2}, 1)$-streaming transducer (Definition 11). To formalise this we introduce probability measures over infinite strings.

A *distribution on finite set* A is a function $d : A \to [0, 1]$ with $\Sigma_{a \in A} d(a) = 1$. For instance, $d : \sigma \mapsto \frac{1}{|A|}$ is a distribution on A, and $d' : \sigma_1 \cdots \sigma_n \mapsto \prod_i d(\sigma_i)$ is a distribution on A^n. The distributions over A will be denoted $\mathtt{dbn}(A)$. For $u \in A^*$, let $\mathtt{cone}(u) \subset A^\omega$ be the set of infinite strings that have u as a prefix. There is a unique *probability measure* μ_d on the Borel σ-field generated by the cones[4] with the property that the measure of $\mathtt{cone}(u)$ is $d'(u)$.

Example 10. Continuing with Example 9, let d and d' be as above (thus $d'(u) := 3^{-|u|}$). The probability (wrt. μ_d) that every prefix of an infinite string is in $(a+b)c^*a^+$ conditioned on the infinite string having a prefix in $R = (a+b)c^*(a^+ + b^+)$ is $\frac{1}{2}$.

Definition 11 ((η, α)-streaming transducer). *Fix regular languages R, T, and a streaming transducer $\mathfrak{Tr}$ from R to T, and a non-negative integer α. Say that infinite string $u = a_0 a_1 a_2 \ldots$*

1. is in need of R-repair *if there exists n so that $a_0 \ldots a_n$ is in R;*
2. *is* $\langle R, T\rangle$-repairable by $\mathfrak{Tr}$ within α *if for all n with $a_0 \ldots a_n \in R$ the cost $\textsc{cost}_{\mathfrak{Tr}}(a_0 \ldots a_n)$ is at most α and $\mathfrak{Tr}(a_0 \ldots a_n) \in T$.*

Say that $\mathfrak{Tr}$ is an (η, α)-streaming transducer (from R to T) *if η is the probability that u is $\langle R, T\rangle$-repairable by $\mathfrak{Tr}$ within α conditioned on u being in need of R-repair. Here probabilities are taken with respect to μ_d induced by the measure on cones d' itself determined by $d : w \mapsto |\Sigma|^{-|w|}$ where Σ is the alphabet of R. When R and T are fixed we may not mention them.*

We give an example where η is close to 1:

Example 12. Let $\Sigma = \{a, b, c, d\}$ and $R_k := (\{a, b\}^k \setminus b^k)c^+ \cup b^k d^+$ and $T = (a+b)^*c^+$. Fix k and note that $\textsc{cost}(R_k \rightsquigarrow T) = \infty$ since $\textsc{ed}(b^k d^n, T) = n$ (as T does not accept any string with d in it). However, there exists an (η, α)-streaming transducer with $\alpha = 0$ and $\eta = 1 - \frac{1}{2^k}$ which operates as follows: it copies the first k letters and then outputs a 'c' for every remaining input letter. The only strings in R_k which it cannot repair within cost 0 are the ones of the form $b^k d^+$.

Partial-repair Problem. The *bounded-repair problem* is, given DFAs for R and T to decide whether or not there exists a streaming transducer $\mathfrak{Tr}$ from R to T such that $\textsc{cost}_{\mathfrak{Tr}}(R)$ is finite. It is proved in [2] that the bounded repair problem is PTIME-complete. Moreover, if it exists, a streaming transducer can be constructed quite easily from their proof.

[4] The Borel σ-field is defined as the least collection of subsets of A^ω containing the cones and closed under countable union and complementation. Sets in the σ-field are called measurable. All our sets in this paper are measurable.

Question 13. Suppose $\text{COST}(R \rightsquigarrow T) = \infty$, or $\text{COST}_{\mathfrak{Tr}}(R)$ is ∞ or just very large for every streaming-transducer $\mathfrak{Tr}$ from R to T. How to transform R to T?

Our proposal is, given R, T and allowed cost α, to construct a streaming transducer $\mathfrak{Tr}$ that, roughly, repairs as many strings as possible. Formally this means solving the *partial-repair problem*: compute the largest η for which there exists a (η, α)-streaming transducer from R to T; and compute the corresponding transducer. The main theorem of this section states that we can do this in PTIME:

Theorem 14 (partial-repair problem). *Given DFAs for R and T, and positive integer α, given in unary, one can compute, in PTIME, the largest $\eta \in [0,1]$ for which there exists an (η, α)-streaming transducer sending R to T.*[5] *Moreover, we can build an (η, α)-streaming transducer from R to T in PTIME.*

The rest of the paper is devoted to a proof of this theorem.

3.1 Tools for Theorem 14

Definition 15 (MC). *A* Markov chain *M is a tuple (Q, Δ, ι) where*

- *Q is a finite set of states,*
- *$\Delta : Q \to \mathtt{dbn}(Q)$ gives the* transition probabilities*, and*
- *$\iota \in \mathtt{dbn}(Q)$ is the* initial distribution.

The edges *E consist of pairs (q, q') such that $\Delta(q)(q') > 0$. A* path *$q_1 q_2 \cdots$* of *M is a (finite or infinite) sequence of states such that $\iota(q_1) > 0$ and successive states q_i, q_{i+1} satisfy E.*

Write Ω_M for the set of infinite paths in M. Form a topology on Ω_M by taking as basis the sets of the form $\mathtt{cone}(x)$ where $\mathtt{cone}(x)$ consists of all infinite paths in M that start with the finite path x. Define the *probability in M of a path* $q_1 \cdots q_n \in Q^+$ as $\iota(q_1) \times \prod_{1 \le i < n} \Delta(q_i)(q_{i+1})$. Define μ_M on $\mathtt{cone}(x)$ as the probability in M of path x. Then μ_M can be uniquely extended to the Borel σ-field generated by the open sets. Write $\mathtt{Pr}_M$ for the unique probability measure (over Ω_M) extending μ_M.

Definition 16 (Labelled MC). *A Markov chain $M = (Q, \Delta, \iota)$ is Σ-*labelled *if for each $q \in Q$ the edges out of q (ie. $E(q) := \{(q, q') : E(q, q')\}$) are in bijection with Σ.*

Being labelled means that every state q has exactly $|\Sigma|$ edges, and each edge goes to a different state.

Example 17 (Uniform MC U_Σ). Let $U = U_\Sigma$ have states Σ, transition from σ to σ' labelled σ' with probability $\frac{1}{|\Sigma|}$, initial distribution sends σ to $\frac{1}{|\Sigma|}$. Then U is a Σ-labelled MC. The probability of $u \in \Sigma^+$ is equal to $|\Sigma|^{-|u|}$. Thus the measure $\mathtt{Pr}_U$ agrees with μ_d on the cones $\mathtt{cone}(u)$. Hence $\mathtt{Pr}_U$ and μ_d agree on the measurable subsets of Σ^ω.

[5] That is, η is a rational and the algorithm computes a representation for it in PTIME.

The following lemma is standard:

Lemma 18. *There is a PTIME algorithm that given a MC M and a set of states A computes the probability that a path in M reaches A.*

The following definition annotates a MC by the states of a DFA.

Definition 19 **($M \oslash_{mc} D$).** *For Σ-labelled Markov chain M and DFA D over alphabet $\Sigma \times Q_M$ define the Σ-labelled Markov chain $M \oslash_{mc} D$ as follows:*

- *The state set is $Q_M \times Q_D$.*
- *Suppose there is an edge $E_M(m, m')$ labelled σ. Then there is an edge from (m, d) to $(m', \delta_D(d, (\sigma, m')))$ with probability $\Delta(m)(m')$ and label σ.*
- *the initial distribution sends (m, d) to $\iota_M(m)$ if d is the initial state of D, and to zero otherwise;*

As a degenerate case, in case D has alphabet Σ then write $M \oslash_{mc} D$ to mean $M \oslash_{mc} F$ where F has the same state set as D, the same initial state, has alphabet $\Sigma \times Q_M$, and sends, for all m, state q on input (σ, m) to $\delta_D(q, \sigma)$.

Note. It can be checked that the object defined is indeed a Σ-labelled Markov chain. We point out that in an edge (m, d) to (m', d') labelled σ, the state d' depends directly on m' — not m — and σ.

Let M and D be as in the definition. Every path $m_1 m_2 \cdots$ of M induces a unique sequence of labels $\sigma_1 \sigma_2 \cdots$ such that the edge from (m_i, m_{i+1}) is labelled σ_i which itself induces a unique sequence $d_1 d_2 \cdots$ of states of D satisfying $d_{i+1} = \delta_D(d_i, (\sigma_i, m_{i+1}))$ where d_1 is the initial state of D. Let

$$\rho : m_1 m_2 \cdots \mapsto (m_1, d_1)(m_2, d_2) \cdots$$

be the *annotation map.* Note that since D is a DFA ρ is a bijection between paths in M and paths in $M \oslash_{mc} D$.

Lemma 20. *Let M be a Σ-labelled MC, D a DFA over $\Sigma \times Q_M$. Then for every measurable $X \subset (Q_M)^\omega$, $\mathit{Pr}_M(X) = \mathit{Pr}_{M \oslash_{mc} D}(\rho(X))$.*

For the proof it is enough to consider X of the form $\texttt{cone}(x)$.

Example 21. Let R be a DFA over Σ with final states F_R and $U = U_\Sigma$ the uniform Markov chain. The lemma says that $\texttt{Pr}_U$ of the set of paths in need of R-repair equals the probability in $\texttt{Pr}_{U \oslash_{mc} R}$ of the set of paths that reach a state of the form $\Sigma \times F_R$.

Definition 22. *A* Markov decision process *is a tuple $((V, E), (V_{dec}, V_{rand}), \mu_\iota, \mu)$ where*

- *(V, E) is a directed graph, V_{dec}, V_{rand} partition V,*
- *$\mu \colon V_{rand} \to \mathtt{dbn}(V_{dec})$ is the edge distribution,*
- *$\mu_\iota \in \mathtt{dbn}(V_{rand})$ is the initial distribution,*
- *for $u \in V_{rand}, (u, v) \in E$ iff $\mu(u)(v) > 0$,*

– *for $v \in V$, $E(v)$ (the out-going edges from v) is non-empty.*

We say that the vertices V_{dec} belong to the ***decider****, while the vertices V_{rand} belong to the* ***randomizer****. A* play *is a path in (V, E).*

We think of a labelled MDP just as a labelled MC with the addition that decider's edges are labelled by elements from a set $\mathtt{Act}$. Formally:

Definition 23 (Labelled MDP). *An MDP is* $(\Sigma, \mathtt{Act})$-labelled *if*

– *for each $v \in V_{rand}$ the edges from v are in bijection with Σ, and*
– *for each $v \in V_{dec}$ each edge from v is labelled by an element of $\mathtt{Act}$.*

Note (identification) Suppose $|E(u)| = 1$ for all $u \in V_{dec}$. Then a $(\Sigma, \mathtt{Act})$-labelled MDP can be naturally viewed as Σ-labelled MC as well as a streaming transducer with input alphabet Σ and output values taken from $\mathtt{Act}$.[6] We call this *identification* and write things like "this MDP is the same, modulo the identification, as that MC".

Definition 24 (Strategy in an MDP). *A* strategy *σ for the decider is a function σ: $V^* \cdot V_{dec} \to V$ such that for all $w \in V^*$ and all $v \in V_{dec}$ we have $\sigma(w \cdot v) \in E(v)$. A* memoryless strategy *for the decider is independent of the history and depends only on the current state, and can be described as a function $\sigma : V_{dec} \to V$. A* finite-state strategy *for the decider is one induced by a DFA (Q, δ, ι) over input alphabet V and output function $\theta : V \times Q \to V$ as follows: $\sigma(wv) := \theta(v, \delta(\iota, w))$.*

As usual, applying a strategy $\mathtt{s}$ to an MDP G results in a MC, which we write $G[\mathtt{s}]$.

Definition 25. *Let $\mathtt{s}$ be a finite-state strategy in G. Write $\mathfrak{Tr}_{\mathtt{s}}$ for the streaming transducer associated with $G[\mathtt{s}]$. Note that $\mathfrak{Tr}_{\mathtt{s}}$ has input alphabet Σ and outputs elements from $\mathtt{Act}$.*

We now define a certain interleaving of a MC and an NFA yielding an MDP. The idea is that the MC determines the allowed moves of the randomizer while the NFA determines the allowed moves of the decider.

Definition 26 ($M \oslash_{mdp} N$). *Suppose M is a Σ-labelled Markov chain and N is an NFA over alphabet $\Sigma \times \mathtt{Act}$. Define a $(\Sigma, \mathtt{Act})$-labelled MDP $M \oslash_{mdp} N$ as follows:*

– *the randomizer's nodes are $Q_M \times Q_N$,*
– *the decider's nodes are $Q_M \times \Sigma \times Q_N$;*
– *if in M there is an edge from m to m' with label σ and probability x, then for all n there is an edge in $M \oslash_{mdp} N$ from (m, n) to (m', σ, n) with label σ and probability x;*

[6] Later $\mathtt{Act}$ will be a set of strings output by a streaming transducer.

– *if in N there is a transition from n to n' labelled $\sigma \in \Sigma$ and $a \in$ `Act`, then for all m there is an edge from (m, σ, n) to (m, n') labelled a;*
– *the initial distribution sends (m, n) to $\iota_M(m)$ if n is the initial state of N, and to 0 otherwise.*

Note. In case D is a DFA then $M \oslash_{mdp} D$ (with the `Act`-labelling removed) is, modulo identification, the Σ-labelled MC $M \oslash_{mc} D$.

Lemma 27. *If $\mathtt{s}$ is a finite-state strategy then the MC $(M \oslash_{mdp} N)[\mathtt{s}]$ is, modulo identification, the same as $M \oslash_{mc} D$ for some DFA D.*

An *objective Φ* for a game graph is a subset of plays. We consider two types of objectives, reachability and safety objectives. Given a set $X \subset V$ of nodes, the reachability objective $\mathtt{reach}(X)$ requires that some vertex in X be visited, and dually, the safety objective $\mathtt{safe}(X)$ requires that only vertices in X be visited. *Solving an MDP for objective Φ* means finding a strategy $\mathtt{s}$ such that, amongst all possible strategies, the probability in the chain $G[\mathtt{s}]$ of Φ is maximised. We call this maximal probability the *value* of the MDP for objective Φ. The following lemma is a slight variation on the standard problem of solving MDPs with reachability objectives.

Lemma 28. *There is a PTIME algorithm that computes the value (and strategy) of an MDP whose objective is a boolean combination of reachability objectives.*

3.2 Proof of Theorem 14

From DFAs R and T and non-negative integer α we construct an MDP $G_{R,T,\alpha}$ and objectives $\mathtt{reach}(\mathtt{Ob}_R)$ and $\mathtt{safe}(\mathtt{Ob}_S)$ such that the following two quantities are equal: 1) the maximum probability over all strategies of $\mathtt{safe}(\mathtt{Ob}_S)$ conditioned on $\mathtt{reach}(\mathtt{Ob}_R)$; 2) the largest $\eta \in [0, 1]$ for which there exists an (η, α)-streaming transducer sending R to T. We now provide the construction (in I), then show how to compute the value (in II), and finally prove that the value is equal to the required conditional probability (in III).

I. Construction of MDP $G_{R,T,\alpha}$. The MDP is constructed in three steps:

Step 1. From the DFA $R = (Q_R, \Sigma, \delta_R, q_{0R}, F_R)$ construct the Σ-labelled Markov chain $UR := U_\Sigma \oslash_{mc} R$, as in Example 21. Its state set is $\Sigma \times Q_R$.

Step 2. From the DFA $T = (Q_T, \Sigma, \delta_T, q_{0T}, F_T)$ and non-negative integer α construct an NFA (without final states) T_α over alphabet $\Sigma \times \Sigma^*$ that simulates the possible repairs of the input string. It does this by storing the allowed number of edit operations left.The non-determinism will corresponds to Decider's choices in the MDP. First we need some notation. Write $\textsc{best-str}(q, q', \sigma)$ for some fixed string w (say the length-lexicographically least) amongst those for which $\textsc{ed}(w, \sigma)$ is minimal with the property that $\delta_T(q, w) = q'$. Now, define T_α as follows:

– the states are $Q_T \times \{\perp, 0, 1, \cdots, \alpha\}$ (here $\perp$ means we have failed to repair the input string);
– the initial state is (q_{0T}, α) (meaning initially there are α edit operations available);
– on input (σ, σ) there is a transition from $(q, \perp)$ to $(\delta_T(q, \sigma), \perp)$, (ie. once we have failed to repair, just copy the input to the output);
– on input $(\sigma, \text{BEST-STR}(q, q', \sigma))$ there is a transition from (q, n) to (q', m) where

$$m = n - \text{ED}(\text{BEST-STR}(q, q', \sigma), \sigma)$$

if this quantity is non-negative, and otherwise $m = \perp$.

Step 3. Define the Σ-labelled MDP $G_{R,T,\alpha}$ as $\oslash_{mdp}$-product of the Markov chain UR and NFA T_α.

Notation. Every node in V_{rand} is of the form $(q, t, n) \in Q_{UR} \times Q_T \times \{\perp, 0, 1, \cdots, \alpha\}$. Every node in V_{dec} is of the form $(q, \sigma', t, n) \in Q_{UR} \times \Sigma \times Q_T \times \{\perp, 0, 1, \cdots, \alpha\}$. The second component of the element $q \in Q_{UR} := \Sigma \times Q_R$ is called the *Q_R-component* of (q, t, n) and of (q, σ', t, n).

Objectives. We introduce two objectives. Let $\texttt{Ob}_R$ be the set of states of $G_{R,T,\alpha}$ whose Q_R-component is in F_R. Let $\texttt{Ob}_S$ be the set of states of $G_{R,T,\alpha}$ such that if the Q_R-component is in F_R then both $t \in F_T$ and $n \neq \perp$. The two objectives are $\texttt{reach}(\texttt{Ob}_R)$ and $\texttt{safe}(\texttt{Ob}_S)$.

II. Computing the Value of $G_{R,T,\alpha}$. Given R,T and α, the value η_* of $G_{R,T,\alpha}$ is defined as the maximum, over all strategies $\texttt{s}$, of the conditional probability,

$$\texttt{Pr}_{G_{R,T,\alpha}[\texttt{s}]}(\texttt{safe}(\texttt{Ob}_S) \mid \texttt{reach}(\texttt{Ob}_R)) = \frac{\texttt{Pr}_{G_{R,T,\alpha}[\texttt{s}]}(\texttt{safe}(\texttt{Ob}_S) \cap \texttt{reach}(\texttt{Ob}_R))}{\texttt{Pr}_{G_{R,T,\alpha}[\texttt{s}]}(\texttt{reach}(\texttt{Ob}_R))}$$

Proposition 29. *The value of the mdp $G_{R,T,\alpha}$ is computable in PTIME and can be realised by a memoryless strategy.*

For the proof observe that the value of $\texttt{reach}(\texttt{Ob}_R)$ is independent of the strategy $\texttt{s}$ chosen by the decider. This is so because $\texttt{s}$ does not have any effect on the Q_R-component of the state of $G_{R,T,\alpha}$. Thus the value of $\texttt{reach}(\texttt{Ob}_R)$ can be easily calculated (fix any memoryless strategy and apply Lemma 18). So, we just need to find the value of the objective $\texttt{safe}(\texttt{Ob}_S) \cap \texttt{reach}(\texttt{Ob}_R)$. By Lemma 28 this can be computed, and the required strategy is memoryless.

III. Existence of an Optimal Streaming Transducer. Fix R, T and α, and let η_* be the value of the MDP $G_{R,T,\alpha}$ for the property $\texttt{safe}(\texttt{Ob}_S)$ conditioned on $\texttt{reach}(\texttt{Ob}_R)$. In this section Proposition 31 immediately implies what we want, namely: 1) there is an (η_*, α)-streaming transducer from R to T; 2) there is no (η, α)-streaming transducer from R to T with $\eta > \eta_*$.

Lemma 30. *Let $\texttt{s}$ be a finite-state strategy for $G_{R,T,\alpha}$ and $\mathfrak{Tr}_{\texttt{s}}$ the corresponding streaming transducer. Then the probability in MC $G_{R,T,\alpha}[s]$ of $\texttt{safe}(\texttt{Ob}_S) \cap \texttt{reach}(\texttt{Ob}_R)$ divided by the probability of $\texttt{reach}(\texttt{Ob}_R)$ is equal to the probability that a string in need of R-repair is $\langle R, T\rangle$-repaired by $\mathfrak{Tr}_{\texttt{s}}$ within α.*

Proof. By Lemma 27 $G_{R,T,\alpha}[\mathtt{s}]$ is a MC of the form $UR \oslash_{mc} D$ for some DFA D. Thus the probability of $\mathtt{reach}(\mathtt{Ob}_R)$ in $G_{R,T,\alpha}[\mathtt{s}]$ is equal to the probability in $UR \oslash_{mc} D$ of reaching a state whose Q_R-component is in F_R. Note that the annotation map ρ preserves the property that a path reaches a state whose Q_R-component is in F_R. By Lemma 20 the latter is equal to the probability in UR that a path reaches a state whose Q_R-component is in F_R. By Example 21 this is equal to the probability that that an infinite string is in need of repair. Similarly it can be shown that the probability of $\mathtt{safe}(\mathtt{Ob}_S)$ in $G_{R,T,\alpha}[\mathtt{s}]$ is equal to the probability in that an infinite string is $\langle R, T\rangle$-repairable by $\mathfrak{Tr}_{\mathtt{s}}$ within α; and the same for the intersection. □

Proposition 31. *1. From a memoryless strategy* $\mathtt{s}$ *in* $G_{R,T,\alpha}$ *with probability (of the conditional objective)* η *one can construct an* (η, α)*-streaming transducer from* R *to* T.

2. From an (η, α)*-streaming transducer* $\mathfrak{Tr}$ *from* R *to* T *one can construct a strategy* $\mathtt{s}_{\mathfrak{Tr}}$ *in* $G_{R,T,\alpha}$ *with value* $\geq \eta$.

Proof. The first item is immediate from Lemma 30.

For the second, a transducer $\mathfrak{Tr}$ gives rise to the following strategy $\mathtt{s}_{\mathfrak{Tr}}$: suppose $\rho \in V^*$ is a play ending in $(q, \sigma, t, n) \in V_{dec}$ where q is a state of UR and t of T_α. Let $\mathtt{in}(\rho)$ be the input (ie. the letters that Randomizer has chosen) and note that it ends in σ. The strategy $\mathtt{s}_{\mathfrak{Tr}}$ sends ρ to node $(q, t', n') \in V_{rand}$, where $t' = \delta_T(q_{0T}, \mathfrak{Tr}(\mathtt{in}(\rho)))$ where δ_T is the transition function and q_{0T} the initial state of the DFA for T. Note that $\mathtt{s}_{\mathfrak{Tr}}$ is a finite-state strategy. So let $\mathfrak{Tr}'$ be the transducer associated with strategy $\mathtt{s}_{\mathfrak{Tr}}$ and apply Lemma 30. Then the probability in $G_{R,T,\alpha}[\mathtt{s}_{\mathfrak{Tr}}]$ of $\mathtt{safe}(\mathtt{Ob}_S) \cap \mathtt{reach}(\mathtt{Ob}_R)$ divided by the probability of $\mathtt{reach}(\mathtt{Ob}_R)$ is equal to the probability that a string in need of R-repair is $\langle R, T\rangle$-repaired by $\mathfrak{Tr}'$ within α. It is required to show that this latter probability is at least η. For this it is sufficient to show that if a string is repaired by $\mathfrak{Tr}$ then it is repaired by $\mathfrak{Tr}'$. But this is the case because although both $\mathfrak{Tr}$ and $\mathfrak{Tr}'$ suggest the same next state for a given input string, say from (q, σ, t, n) to (q, t', n'), they possibly differ on the output string, say u and v. In particular, $\textsc{ed}(\sigma, u) \geq \textsc{ed}(\sigma, v) := \textsc{best-str}(t, t', \sigma)$ by the definition of $G_{R,T,\alpha}$. □

References

1. Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions, cost register automata, and generalized min-cost problems. *CoRR*, abs/1111.0670, 2011.
2. Michael Benedikt, Gabriele Puppis, and Cristian Riveros. Regular repair of specifications. In *LICS*, pages 335–344, 2011.
3. Robert A. Wagner. Order-n correction for regular languages. *Commun. ACM*, 17(5):265–268, may 1974.