Weighted Automata Computation of Edit Distances with Consolidations and Fragmentations

Mathieu Giraud, Florent Jacquemard

Weighted Automata Computation of Edit Distances with Consolidations and Fragmentations

Mathieu Giraud[1] and Florent Jacquemard[2]

[1]CRIStAL, UMR 9189 CNRS, Univ. Lille, France.
mathieu@algomus.fr
[2]INRIA, Paris, France. florent.jacquemard@inria.fr

August 14, 2018

Abstract

We study edit distances between strings, based of based on operations of character substitutions, insertions, deletions and additionally *consolidations* and *fragmentations*. The two latter operations transform a sequence of characters into one character and vice-versa. They correspond to the compression and expansion in Dynamic Time-Warping algorithms for speech recognition and are also used for the formal analysis of written music.

Such edit distances are not computable in general. We propose weighted automaton constructions to compute in polynomial time an edit distance taking into account both consolidations and deletions, or both fragmentations and insertions. We finally show that the optimal weight of sequences made of consolidations chained with fragmentations, in that order, is computable in polynomial time, and not computable if we reverse the order of fragmentations and consolidations.

1 Introduction

Edit distances measure similarity between two sequences of symbols, usually with substitution, insertion and deletion *operations*. Each of these operations is given a *weight* (or *cost*), and the edit distance is defined as the minimal weight of a transformation of one sequence into the other using the operations. For the Levenshtein distance [15], all operations have a weight of 1, but more elaborated weights schemes are also used: In bioinformatics, to compare proteins seen as sequences over the 20-letter alphabet of amino acids, the BLOSUM62 substitution matrix [4] reflects the probability of substitution of any two amino acids. Edit distances are also applied in computer music to measure melodic similarity. This is particularly useful in a digital

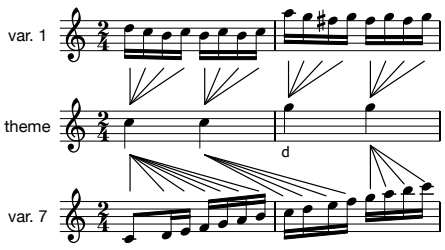

Figure 1: Theme, Variations 1 and 7 from Mozart's *"Ah vous dirai-je maman"* Variations K265. Transformations between the theme and the variations can be seen as a sequence of *fragmentation* operations, breaking a note into several ones and possibly substituting resulting notes, with also a deletion for the Variation 7.

humanities context, for assistance to the analysis of written music. In 1990, Mongeau and Sankoff [12] proposed to extend edit distance with *consolidation* and *fragmentation* operations to account for common transformations between melodic patterns – see Figure 1. These operations correspond to the *compression* and *expansion* in dynamic time-warping algorithms for *e.g.* speech recognition [7].

Edit distances between strings are usually computed with dynamic programming [15]. More generic algorithms have been proposed to compute the edit distance between two regular languages [11] with construction and composition of weighted transducers. These approaches are applied to standard Levenshtein edit-distance and are applicable to more general edit-distance defined by arbitrary sets of edit operations, provided that these operations are expressed as weighted string transducers [11]. In an unweighted context, it has been shown that some class of string rewriting rules including consolidation (but much more general) effectively preserves regularity [5]. This means that given a finite automaton $\mathcal{A}$ and such a set of string rewriting operations E, it is possible to compute a finite automaton recognising the closure of the language of $\mathcal{A}$ by E.

In this paper, after defining edit distances with consolidation and fragmentation operations, we show that they are not computable in general (Section 2). We then propose a construction of weighted automata to compute the optimal weight of edit operations sequences including usual operations *together with consolidations*, or either *with fragmentations*, but not both in the same time (Section 3). We finally show how to account for edit operations sequences that include consolidations *chained* with fragmentations (Section 4).

2 Edit Distance with Consolidations and Fragmentations

In this section, we define extended edit distances between finite sequences of symbols, that can include generic *consolidation* or *fragmentation* operations

and where the weights are values in a semiring with some properties.
Let us consider a fixed finite alphabet Σ. We use letters a, b... to denote symbols of Σ. The monoid of words (strings) over Σ is denoted Σ^*, ε is the empty sequence and uv denotes the concatenation of $u \in \Sigma^*$ and $v \in \Sigma^*$. For a string $s = a_1 \dots a_n \in \Sigma^*$, and $0 \leq i \leq n$, $s_{1..i}$ denotes the prefix of s of length i: $s_{1..i} = a_1 \dots a_i$ when $0 < i \leq n$, and $u_{1..0} = \varepsilon$.

2.1 Edit Operations and Rulesets

The Levenshtein edit distance [15] between two strings s and t is defined as the minimum number of elementary editions transforming s into t, using operations of substitution (replacement of one character by another), single-character insertion and deletion. Here, we consider more general edition operations ranged in two families, and which include substitution, insertion and deletion.

An *edit operation* is a pair $u \to v$ with $u, v \in \Sigma^*$. Note that both u and v can be ε, and that this is generally not the case for u in the string rewriting theory [1]. The operations of the first family, called *fragmentations*, have the form

$$a \to b_1 \dots b_n \qquad \text{(fragmentation)}$$

where $a, b_1, ..., b_n \in \Sigma$ and $n \geq 0$. When $n = 0$, the fragmentation operation is called (deletion), and when $n = 1$, it is a (substitution). The operations of the second family, called *consolidations*, have the form

$$a_1 \dots a_n \to b \qquad \text{(consolidation)}$$

where $a_1, ..., a_n, b \in \Sigma$ and $n \geq 0$. When $n = 0$, the consolidation operation is called (insertion). The case $n = 1$ still corresponds to the above case of substitution. A *ruleset* E is a finite set of edit operations. Its *size* $\|E\|$ is the sum of the size of the strings in its rules.
Its *symmetric* is $E^{-1} = \{v \to u \mid u \to v \in E\}$.

Example 1. *The ruleset* $E_0 = \left\{a \to b, \varepsilon \to b, a \to \varepsilon \;\middle|\; a, b \in \Sigma, a \neq b\right\}$ *contains all the substitution, insertion, and deletion operations over* Σ.

Example 2. *Figure 2 describes, on a 3-letter alphabet, the rulesets* $E_0 = E_{3,\mathsf{s}} \cup E_{3,\mathsf{i}} \cup E_{3,\mathsf{d}}$ *(substitutions, insertions, deletions),* $E_{3,\mathsf{c}}$ *(same-letter consolidations up to 3 letters) and* $E_{3,\mathsf{f}}$ *(same-letter fragmentations up to 3 letters),*

In music analysis, consolidation/fragmentation rules are used to transform a note into several ones, typically preserving the total duration (See Figure 1). In speech processing, compression and expansion are used in time-wrapping techniques for comparing time series subject to variations in speed [7]. In both of these cases, it is preferable to define these operations rather than using several insertions/deletions.

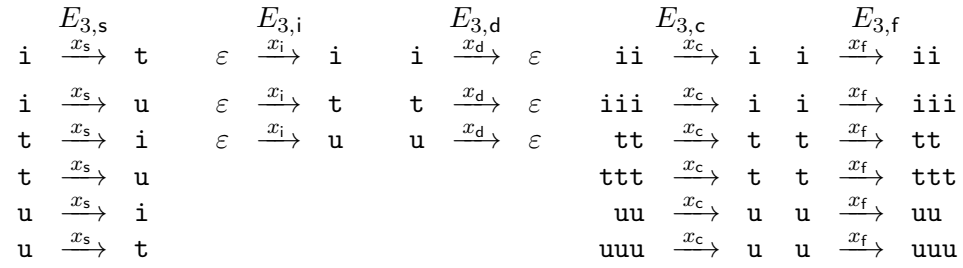

Figure 2: Rulesets over the alphabet $\Sigma_3 = \{\mathtt{i}, \mathtt{t}, \mathtt{u}\}$ for substitutions ($E_{3,\mathtt{s}}$, weight $x_\mathtt{s}$), insertions ($E_{3,\mathtt{i}}$, $x_\mathtt{i}$), deletions ($E_{3,\mathtt{d}}$, $x_\mathtt{d}$), consolidations and fragmentations up to 3 letters ($E_{3,\mathtt{c}}$ and $E_{3,\mathtt{f}}$, of respective weights $x_\mathtt{c}$ and $x_\mathtt{f}$).

2.2 Weight Domains

An edit distance is defined by assigning every edit operation in a ruleset E with a weight value. In the case of usual edit distances based on E_0, these weight values are strictly positive real numbers (1 for the Levenshtein distance) and the weight of a sequence of edit operations is the sum of the individual weights of the rules involved. The edit distance between two strings s and t is then the minimal weight of a sequence transforming s into t.

We consider here abstract domains for weight values, as semirings with an operator $\otimes$, for the composition of weights in a sequence of edit operations, and an operator $\oplus$ for the selection of the optimal sequence in the definition of the edit distance. This generalisation simplifies the proofs and enables identification of the algebraic properties of the weight domain which are necessary for ensuring the correctness of automata construction for the computation of edit distances.

Definition 1. *A* semiring $\mathcal{S} = \langle \mathbb{S}, \oplus, \otimes, \mathbb{0}, \mathbb{1} \rangle$ *is a structure with a domain* $\mathbb{S} = dom(\mathcal{S})$ *equipped with two binary operators* $\oplus$ *and* $\otimes$ *such that:* $\langle \mathbb{S}, \oplus, \mathbb{0} \rangle$ *is a commutative monoid:* $\oplus$ *is associative and commutative and* $\mathbb{0} \in \mathbb{S}$ *is a neutral element for* $\oplus$*;* $\langle \mathbb{S}, \otimes, \mathbb{1} \rangle$ *is a monoid:* $\otimes$ *is associative and* $\mathbb{1} \in \mathbb{S}$ *is a neutral element for* $\otimes$*;* $\otimes$ *distributes over* $\oplus$*:* $\forall x, y, z \in \mathbb{S}$*,* $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ *and* $(x \oplus y) \otimes z = (x \otimes z) \oplus (y \otimes z)$*; and* $\mathbb{0}$ *is absorbing wrt* $\otimes$*:* $\forall x \in \mathbb{S}$*,* $\mathbb{0} \otimes x = x \otimes \mathbb{0} = \mathbb{0}$*.*

We may simply write $x \in \mathcal{S}$ to mean $x \in \mathbb{S}$. A semiring $\mathcal{S}$ is *commutative* if $\otimes$ is commutative. It is *idempotent* if for all $x \in dom(\mathcal{S})$, $x \oplus x = x$. It is k-closed if for all $x \in dom(\mathcal{S})$, $\bigoplus_{n=0}^{k+1} x^n = \bigoplus_{n=0}^{k} x^n$. Following [10], if $\mathcal{S}$ is 0-closed, *i.e.* when $\forall x \in dom(\mathcal{S}), \mathbb{1} \oplus x = \mathbb{1}$ then it is called *bounded*. Note

semiring	domain	$\oplus$	$\otimes$	$\mathbb{0}$	$\mathbb{1}$	natural ordering
Boolean	$\{0,1\}$	$\vee$	$\wedge$	0	1	$\mathbb{1} \leq_{\mathcal{S}} \mathbb{0}$
Viterbi	$[0,1] \subset \mathbb{R}_+$	max	.	0	1	$x \leq_{\mathcal{S}} y \iff x \geq y$
Tropical	$\mathbb{R}_+ \cup \{+\infty\}$	min	$+$	$+\infty$	0	$x \leq_{\mathcal{S}} y \iff x \leq y$

Figure 3: These common semirings are commutative, idempotent and bounded.

that every bounded semiring is idempotent (by boundedness, $\mathbb{1} \oplus \mathbb{1} = \mathbb{1}$, and idempotency follows by multiplying both sides by x).
A semiring $\mathcal{S}$ is *complete* when every infinite (countable) sum of elements of $dom(\mathcal{S})$ is well-defined and in $dom(\mathcal{S})$, and the following properties hold for every $I \subset \mathbb{N}$:

associativity: for every partition $(I_j)_{j \in J}$ of I, $\bigoplus_{j \in J} \bigoplus_{i \in I_j} x_i = \bigoplus_{i \in I} x_i$

commutativity: for all permutation I' of I, $\bigoplus_{i \in I'} x_i = \bigoplus_{i \in I} x_i$

distributivity of product over infinite sum: for all I,
$\bigoplus_{i \in I} (x \otimes y_i) = x \otimes \bigoplus_{i \in I} y_i$, and $\bigoplus_{i \in I} (x_i \otimes y) = (\bigoplus_{i \in I} x_i) \otimes y$.

Every idempotent semiring $\mathcal{S}$ induces an ordering relation $\leq_{\mathcal{S}}$ called *natural ordering* of $\mathcal{S}$ and defined as, for all x and y, $x \leq_{\mathcal{S}} y$ iff $x \oplus y = x$. In that case, $\mathcal{S}$ is *monotonic wrt* $\leq_{\mathcal{S}}$: for all x, y, z, $x \leq_{\mathcal{S}} y$ implies $x \oplus z \leq_{\mathcal{S}} y \oplus z$, $x \otimes z \leq_{\mathcal{S}} y \otimes z$, and $z \otimes x \leq_{\mathcal{S}} z \otimes y$.

From now on, we assume that $\mathcal{S}$ is a commutative semiring, which is complete and bounded (hence idempotent). This is the case for the three common semirings shown on Figure 3.

2.3 Weighted Edit Operations, Optimal Weight and Edit Distance

A *weighted ruleset* E is a ruleset E equipped with a *weight function*, which is a mapping $\mathsf{weight} : E \to \mathcal{S} \setminus \{0\}$ such that the triangle inequality holds: for all $u, v, w \in \Sigma^*$ such that $u \to v \in E$, $v \to w \in E$, and $u \to w \in E$,

$$\mathsf{weight}(u \to w) \leq_{\mathcal{S}} \mathsf{weight}(u \to v) \otimes \mathsf{weight}(v \to v) \tag{1}$$

By abuse of notation, we write $u \xrightarrow{x} v \in E$ when $x = \mathsf{weight}(u \to v)$. A weighted ruleset E is *symmetric* when $E = E^{-1}$ and for every operation $u \to v \in E$, $\mathsf{weight}(v \to u) = \mathsf{weight}(u \to v)$.

A *positioned operation* π of E is a pair $\langle u \to v, i\rangle$ made of an edit operation of E and a natural number, meaning that $u \to v$ is applied at position i in a string. We write $s \xrightarrow{\pi} t$ when for some $w, w' \in \Sigma^*$, $s = wuw'$, $t = wvw'$, and $i = |w|$. For all $s, t \in \Sigma^*$, we write $s \xrightarrow[E]{} t$ when $s \xrightarrow{\langle u \to v, i\rangle} t$ for some $u \to v \in E$, and $s \xrightarrow[E]{*} t$ for the transitive closure of this relation, called *edition*.

Let $\sigma = \pi_1\pi_2 \dots \pi_n$ be a finite sequence of positioned operations of E (called *edition sequence*), with $\pi_j = \langle u_j \to v_j, i_j\rangle$ for all $j = 1..n$. We write $s \xrightarrow{\sigma} t$ if $s = s_0 \xrightarrow{\pi_1} s_1 \dots \xrightarrow{\pi_n} s_n = t$ for some $s_0, \dots, s_n$. The weight of the sequence σ is defined by

$$\mathsf{weight}(\sigma) = \begin{cases} \mathbb{1} & \text{if } \sigma \text{ is empty } (n = 0), \\ \bigotimes_{i=1}^{n} \mathsf{weight}(u_i \to v_i) & \text{otherwise.} \end{cases}$$

Then we define the *optimal weight* between s and t *wrt* E by

$$D_E(s,t) = \begin{cases} \mathbb{1} & \text{if s = t,} \\ \bigoplus_{s \xrightarrow{\sigma} t} \mathsf{weight}(\sigma) & \text{otherwise.} \end{cases}$$

When $s \neq t$ and the above sum is empty, then $D_E(s,t) = \mathbb{0}$. Note that the sum may be infinite, hence the hypothesis of completeness of $\mathcal{S}$ is important in our context. In general, we use the term *optimal weight* fo D_E as this measure may not be a distance. When E is symmetric, D_E is a distance called *edit distance*.

Example 3. *On the tropical semiring and assuming that all edit operations have a weight of* $x_{\mathtt{s}} = x_{\mathtt{i}} = x_{\mathtt{d}} = x_{\mathtt{c}} = x_{\mathtt{f}} = 1$, D_{E_0} *is the Levenshtein distance, and* $D_{E_0}(\mathtt{tutti}, \mathtt{ti}) = 3$ *(three deletions), whereas* $D_{E_0 \cup E_{3,\mathtt{c}}}(\mathtt{tutti}, \mathtt{ti}) = 2$ *(with one deletion of* `u` *followed by one consolidation* $\mathtt{ttt} \to \mathtt{t}$*).*

2.4 Computability of D_E

Proposition 2. *In general,* $D_E(s,t)$ *is not computable for a ruleset* E *containing fragmentation and consolidation rules.*

Proof. By definition, the problem whether $D_E(s,t) \neq \mathbb{0}$ for two strings s and t is equivalent to the problem of reachability $s \xrightarrow[E]{*} t$, which is undecidable when E is a set of fragmentation and consolidation rules. This can be shown by reduction of *e.g.* the Post Correspondence Problem (PCP).

Let us consider an instance of PCP $\mathcal{P} = \left\{\langle u_i, v_i\rangle \;\middle|\; i \leq n,\, u_i, v_i \in \Sigma^*\right\}$. The problem is to find a sequence $i_1, \dots, i_k \leq n$ such that $u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$. Let us add two marker symbols $\sharp$ and $\natural$ to Σ.

Intuitively in the reduction of PCP, the fragmentation operations are used for generating a candidate solution (to the problem) from a constant

symbol $\sharp$, and the consolidation operations are used for checking that a candidate is indeed a solution, by reduction to another constant symbol $\natural$. More precisely, let the ruleset E_{f} contain one fragmentation $\sharp \to \widetilde{u_i} \sharp v_i$ for each pair $\langle u_i, v_i \rangle \in \mathcal{P}$ ($\widetilde{u_i}$ is the mirror image of u_i). And let the ruleset E_{c} contain two consolidations $a \sharp a \to \natural$ and $a \natural a \to \natural$ for each $a \in \Sigma$. We assign to all these edit operations a non-null weight in $\mathcal{S}$, say $\mathbb{1}$, and call $E = E_{\mathsf{f}} \cup E_{\mathsf{c}}$.

Then it holds that $D_E(\sharp, \natural) \neq \mathbb{0}$ iff $\sharp \xrightarrow[E]{*} \natural$ iff $\mathcal{P}$ has a solution. □

When E is E_0, D_{E_0} can be computed using dynamic programming equations [13, 15, 14] by tabulating $\Delta(i, j) = D_E(s_{1..i}, t_{1..j})$ with $\Delta(0, 0) = \mathbb{1}$ and

$$\Delta(i,j) = \bigoplus \begin{cases} \Delta(i-1,j-1) & \text{if } s_i = t_j \text{ (match)} \\ \Delta(i-1,j-1) \otimes \mathsf{weight}(s_i \to t_j) & \text{if } s_i \neq t_j \text{ (substitution)} \\ \Delta(i-1,j) \otimes \mathsf{weight}(s_i \to \varepsilon) & \text{(deletion)} \\ \Delta(i,j-1) \otimes \mathsf{weight}(\varepsilon \to t_j) & \text{(insertion)} \end{cases} \tag{dp}$$

Adding lines for fragmentations and consolidations to this equation, as what was done by [12], do not always compute the optimal weight of edit operations sequences, such as the correct value of $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{ti})$ (see Example 3). The following section proposes a construction of a weighted automaton to correctly compute the optimal weight $D_E(s, t)$ for some rulesets E larger than E_0.

3 Weighted Automata Algorithm Computing D_E

Intuitively, $D_E(s, t)$ is the smallest weight (wrt $\leq_{\mathcal{S}}$) of a path in the edit graph associated to E. More precisely, we can associate to E and $s \in \Sigma^*$ a graph $\mathcal{G}_{E,s}$ whose vertices are the descendants of s wrt E $\{w \mid s \xrightarrow{*} w\}$ and where each edge describes one positioned operation of E with its associated weights. $D_E(s, t)$ is then the smallest weight of a path between s and t in $\mathcal{G}_{E,s}$. However, we cannot simply construct $\mathcal{G}_E$ and apply a shortest-path algorithm, like in [10], to compute the optimal weight, because $\mathcal{G}_E$ is not finite in general (consider *e.g.* the case where E contains insertions). Instead, we use as a finite representation of $\mathcal{G}_E$ a weighted automaton $\mathcal{A}_s^E$ computing the optimal weight.

3.1 Weighted Automata

The purpose of a weighted automaton [3] over the alphabet Σ and the semiring $\mathcal{S}$ is to define a mapping from the strings of Σ^* into weights of $\mathcal{S}$.

Definition 3. *A weighted automaton $\mathcal{A}$ over the alphabet Σ and the semiring $\mathcal{S}$ is a tuple $\langle Q, \mathsf{in}, \mathsf{weight}, \mathsf{out} \rangle$ where Q is a finite set of states,* $\mathsf{in} : Q \to$

$\mathcal{S}$, *resp.* $\mathsf{out} : Q \to \mathcal{S}$ *is a function defining the weight for entering, resp. leaving, a state, and* $\mathsf{weight} : Q \times \Sigma \times Q \to \mathcal{S}$ *is a transition weight function.*

The functions defining $\mathcal{A}$ might be subscripted by $\mathcal{A}$ when needed. A state $q \in Q$ such that $\mathsf{in}(q) \neq \mathbb{0}$, resp. $\mathsf{out}(q) \neq \mathbb{0}$, is called *initial*, resp. *final*. The set of *transitions* of $\mathcal{A}$ is $\{\langle q, a, q' \rangle \mid q, q' \in Q, a \in \Sigma\}$. The *size* of $\mathcal{A}$ is $\|\mathcal{A}\| = |Q| + |\{\langle q, a, q' \rangle \mid \mathsf{weight}(q, a, q') \neq \mathbb{0}\}|$. We denote the *source* of a transition $\tau = \langle q, a, q' \rangle$ by $\mathsf{src}(\tau) = q$, its *target* by $\mathsf{trg}(\tau) = q'$ and its *label* by $\mathsf{lab}(\tau) = a$. By abuse, we shall write τ as $q \xrightarrow[\mathcal{A}]{a,x} q'$ when $x = \mathsf{weight}(\tau)$. $\mathcal{A}$ may be omitted when clear from context.

A *run* of the weighted automaton $\mathcal{A}$ is a finite sequence $\rho = \tau_1 \dots \tau_n$ of transitions, with $n > 0$ and such that for all $1 \leq i < n$, $\mathsf{trg}(\tau_i) = \mathsf{src}(\tau_{i+1})$. We write $\mathsf{src}(\rho) = \mathsf{src}(\tau_1)$ and $\mathsf{trg}(\rho) = \mathsf{trg}(\tau_n)$. The run ρ is called a *cycle* when $\mathsf{src}(\rho) = \mathsf{trg}(\rho)$. The length of the run ρ is $\mathsf{len}(\rho) = n$, its *label* is the concatenation $\mathsf{lab}(\rho) = \mathsf{lab}(\tau_1) \dots \mathsf{lab}(\tau_n)$ (we also say that ρ is a *run over* $\mathsf{lab}(\rho)$) and its *weight* is $\mathsf{weight}(\rho) = \bigotimes_{i=1}^{n} \mathsf{weight}(\tau_i)$.

For a string $u \in \Sigma^*$ and $q, q' \in Q$, we denote by $R_{\mathcal{A}}(q, u, q')$ the set of all runs ρ of $\mathcal{A}$ such that $\mathsf{src}(\rho) = q$, $\mathsf{trg}(\rho) = q'$ and $\mathsf{lab}(\rho) = u$, and define

$$\mathsf{weight}_{\mathcal{A}}(q, u, q') = \bigoplus_{\rho \in R_{\mathcal{A}}(q,u,q')} \mathsf{weight}(\rho).$$

When $u = \varepsilon$, $R_{\mathcal{A}}(q, u, q') = \emptyset$, and we define by convention $\mathsf{weight}_{\mathcal{A}}(q, \varepsilon, q) = \mathbb{1}$ for all $q \in Q$ and $\mathsf{weight}_{\mathcal{A}}(q, \varepsilon, q') = \mathbb{0}$ for all $q, q' \in Q, q \neq q'$. Then, the output of $\mathcal{A}$ on $u \in \Sigma^*$ is the following. It can be computed in $O(|u| \cdot \|\mathcal{A}\|)$.

$$\begin{aligned} \mathcal{A}(u) &= \bigoplus_{q,q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_{\mathcal{A}}(q, u, q') \otimes \mathsf{out}(q') \\ &= \bigoplus_{q,q' \in Q} \bigoplus_{\rho \in R_{\mathcal{A}}(q,u,q')} \mathsf{in}(q) \otimes \mathsf{weight}(\rho) \otimes \mathsf{out}(q'). \end{aligned}$$

3.2 Generalised Levenshtein Automaton Construction

The goal of this section is to show that given a ruleset E containing only consolidation and deletion edit operations and a string $s \in \Sigma^*$, one can compute a weighted automaton $\mathcal{A}_s^E$ over Σ and $\mathcal{S}$ that computes $D_E(s, t)$, *i.e.* such that $\forall t \in \Sigma^*, \mathcal{A}_s^E(t) = D_E(s, t)$.
The construction starts with an automaton $\mathcal{A}_s^0$ such that $\mathcal{A}_s^0(s) = \mathbb{1}$ and updates its transition weight function in order to compute the optimal weight.

Example 4. *Figure 4 shows an example of* $\mathcal{A}_s^0$ *for* $s = \texttt{tutti}$ *in the Tropical semiring (where* $\mathbb{1} = 0$*).*

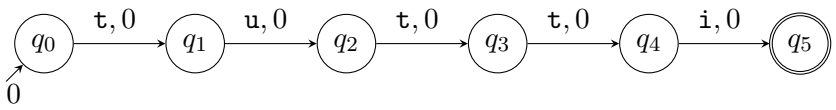

Figure 4: The weighted automaton $\mathcal{A}^0_{\texttt{tutti}}$ on the tropical semiring.

Initial Automaton Let $s = s_1 \dots s_m \in \Sigma^*$ and let us define a weighted automaton over Σ and $\mathcal{S}$

$$\mathcal{A}^0_s = \big(Q = \{q_0, q_1, \dots, q_m\}, \mathsf{in}_0, \mathsf{out}_0, \mathsf{weight}_0\big)$$

where:

$\mathsf{in}_0(q_0) = \mathbb{1}$ and $\mathsf{in}_0(q_i) = \mathbb{0}$ for all $0 < i \leq m$,

$\mathsf{out}_0(q_i) = \mathbb{0}$ for all $0 \leq i < m$ and $\mathsf{out}_0(q_m) = \mathbb{1}$,

$\mathsf{weight}_0(q_i, s_{i+1}, q_{i+1}) = \mathbb{1}$ for all $0 \leq i < m$, and

$\mathsf{weight}_0(q_i, s_j, q_k) = \mathbb{0}$ for all other transitions.

The following lemma is an immediate consequence of the definition of $\mathcal{A}^0_s$.

Lemma 4. *It holds that $\mathcal{A}^0_s(s) = \mathbb{1}$ and $\mathcal{A}^0_s(u) = \mathbb{0}$ for all $u \neq s$.*

Automata Transformation The weighted automaton construction starts with $\mathcal{A}^0_s$, and the state set Q is not changed but the weight functions are updated, in a way that the obtained weighted automaton $\mathcal{A}$ satisfies the following properties:

(c_1) For any *consolidation* $a_1 \dots a_n \xrightarrow{x} b \in E$ with $a_1, \dots, a_n, b \in \Sigma$ and $n \geq 0$,
for all states $q_i, q_j \in Q$, with $0 \leq i \leq j \leq m$,

$$\mathsf{weight}_{\mathcal{A}}(q_i, b, q_j) \leq_{\mathcal{S}} \mathsf{weight}_{\mathcal{A}}(q_i, a_1 \dots a_n, q_j) \otimes x.$$

(c_2) For any *deletion* $a \xrightarrow{x} \varepsilon \in E$ with $a \in \Sigma$, for all states $q_\ell, q_j \in Q$ with $0 \leq \ell \leq j \leq m$,

$$\mathsf{in}_{\mathcal{A}}(q_j) \leq_{\mathcal{S}} \mathsf{in}_{\mathcal{A}}(q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, a, q_j) \otimes x.$$

and for any state $q_i \in Q$ with $0 \leq i \leq \ell$,

$$\mathsf{weight}_{\mathcal{A}}(q_i, b, q_j) \leq_{\mathcal{S}} \mathsf{weight}_{\mathcal{A}}(q_i, b, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, a, q_j) \otimes x.$$

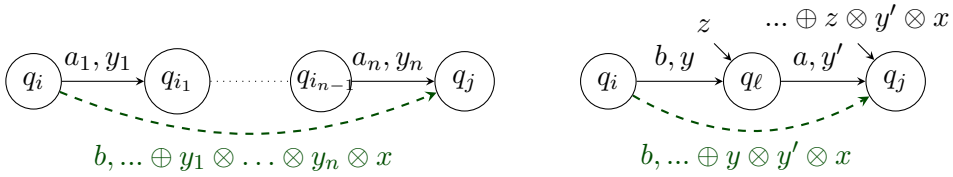

Figure 5: An illustration of the conditions (c_1) (left) and (c_2) (right).

Figure 5 illustrates the intuitions behind these weight updates. The case (c_1) expresses that $\mathcal{A}$ must take into account the possibility to edit $a_1 \ldots a_n$ into b. To that end, the weight of the transition $q_i \xrightarrow{b} q_j$ is updated with the weight x of the consolidation. The case (c_2) corresponds to the addition of an ε-transition $q_\ell \xrightarrow{\varepsilon} q_j$ in place of the transition labeled by a, and the on-the-fly suppression of this ε-transition, following a technique similar to *e.g.* [9, 8].

The updated weights in the automaton transitions are computed using recursive equations over $\mathcal{S}$.

Example 5. *Figure 6 shows an example of the construction of* $\mathcal{A}^E_{\texttt{tutti}}$ *by updating the transition weights of* $\mathcal{A}^0_{\texttt{tutti}}$ *(in the Tropical semiring).*

For all $0 \leq i \leq j \leq m$ (remember that $m = |Q| - 1$ is the length of s) and $a \in \Sigma$, we consider a variable $\mathsf{w}_{i,a,j}$ representing $\mathsf{weight}_{\mathcal{A}}(q_i, a, q_j)$, and a variable y_i representing $\mathsf{in}_{\mathcal{A}}(q_i)$. We call $\mathcal{S}$-*expression* an expression made of the binary operators of $\mathcal{S}$, elements in $dom(\mathcal{S})$ and the above variables. The following equations associate an $\mathcal{S}$-expression to each variable.

$$\begin{aligned}\mathsf{w}_{i,b,j} \quad = \mathsf{weight}_0(q_i, b, q_j) \quad &\oplus \bigoplus_{a_1 \ldots a_n \xrightarrow[E]{x} b} \bigoplus_{\substack{(i_0,\ldots,i_n) \\ i_0=i,\, i_n=j}} x \otimes \bigotimes_{p=0}^{n-1} \mathsf{w}_{i_p,a_{p+1},i_{p+1}} \\ &\oplus \bigoplus_{a \xrightarrow[E]{x} \varepsilon} \bigoplus_{i \leq \ell \leq j} x \otimes \mathsf{w}_{i,b,\ell} \otimes \mathsf{w}_{\ell,a,j}\end{aligned} \tag{e_{12}}$$

$$\mathsf{y}_j \quad = \quad \mathsf{in}_0(q_j) \oplus \bigoplus_{a \xrightarrow[E]{x} \varepsilon} \bigoplus_{\ell \leq j} x \otimes \mathsf{y}_\ell \otimes \mathsf{w}_{\ell,a,j} \tag{e'_2}$$

The first line in (e_{12}) corresponds to the case (c_1) (consolidation) and the second line corresponds to second part of the case (c_2) (deletion) – the inequality concerned with $\mathsf{weight}_{\mathcal{A}}(q_i, b, q_j)$. The equation ($\mathsf{e}'_2$) corresponds to first part of the case (c_2) (deletion) – the inequality concerned with $\mathsf{in}(q_j)$. An *assignation* ϕ is a mapping

$$\phi : \{\mathsf{w}_{i,a,j} \mid 0 \leq i \leq j \leq n, a \in \Sigma\} \cup \{\mathsf{y}_i \mid 0 \leq i \leq n\} \to \mathcal{S}.$$

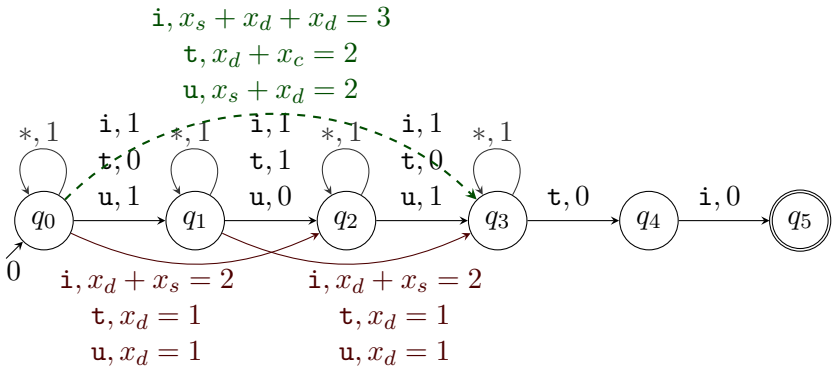

Figure 6: Automaton $\mathcal{A}^E_{\texttt{tutti}}$ on the tropical semiring after solving $\mathsf{w}_{i,a,j}$ for every $a \in \Sigma_3$ and every i and j with $0 \leq i \leq j \leq 3$, considering the ruleset $E_0 \cup E_{3,c}$. All edition rules have here a weight of 1. Solving $\mathsf{w}_{0,a,3}$ variables is done by the case (e_{12})**.2**. For example, $\mathsf{w}_{0,\mathtt{t},3} = \min(+\infty,\, e,\, g + \mathsf{w}_{0,\mathtt{t},3})$, where $e = \min(\mathsf{w}_{0,\mathtt{t},1} + \mathsf{w}_{1,\mathtt{t},2} + x_{\mathtt{c}}, ...)$ contains the weight of the best path and $g = \min(\mathsf{w}_{3,\mathtt{t},3} + x_{\mathtt{d}}, ...)$ corresponds to cycles that will not improve the weight.

We consider the homomorphic extension of an assignation ϕ, also denoted ϕ, mapping $\mathcal{S}$-expressions to $\mathcal{S}$, defined by $\phi(e \oplus e') = \phi(e) \oplus \phi(e')$, $\phi(e \otimes e') = \phi(e) \otimes \phi(e')$ and $\phi(x) = x$ for all $x \in dom(\mathcal{S})$. We associate to an assignation ϕ the weighted automaton $\mathcal{A}_\phi = (Q, \mathsf{in}_\phi, \mathsf{out}_0, \mathsf{weight}_\phi)$ where for all $0 \leq i \leq j \leq m$ and $a \in \Sigma$, $\mathsf{weight}_\phi(q_i, a, q_j) = \phi(\mathsf{w}_{i,a,j})$ and $\mathsf{in}_\phi(q_i) = \phi(\mathsf{y}_i)$.

Lemma 5. *If ϕ is a solution of (e_{12}), (e'_2), then $\mathcal{A}_\phi$ satisfies (c_1) and (c_2).*

Proof. Let $\mathcal{A} = \mathcal{A}_\phi$ (to simplify notations).
For (c_1), if $\mathsf{weight}_{\mathcal{A}}(q_i, b, q_j) = \phi(\mathsf{w}_{i,b,j}) = \mathbb{1}$, then we are done by boundedness of $\mathcal{S}$.
Otherwise, for $a_1 \dots a_n \xrightarrow{x} b \in E$, the sum $\bigoplus_{(i_0,\dots,i_n)} x \otimes \bigotimes_{p=0}^{n-1} \phi(\mathsf{w}_{i_p,a_{p+1},i_{p+1}})$ in (e_{12}) is equal to $x \otimes \mathsf{weight}_{\mathcal{A}}(q_i, a_1 \dots a_n, q_j)$ by definition of $\mathcal{A}_\phi$ and $\mathsf{weight}_{\mathcal{A}}$, and the inequality of (c_1) follows, using commutativity of $\mathcal{S}$.
For (c_2) (case of deletion $b \xrightarrow{x} \varepsilon$), the principle is the same:

$$\bigoplus_{i \leq \ell \leq j} x \otimes \phi(\mathsf{w}_{i,b,\ell}) \otimes \phi(\mathsf{w}_{\ell,a,j}) = x \otimes \mathsf{weight}_{\mathcal{A}}(q_i, b, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, a, q_j)$$

and the second inequality of (c_2) follows;

$$x \otimes \phi(\mathsf{y}_\ell) \otimes \phi(\mathsf{w}_{\ell,a,j}) = x \otimes \mathsf{in}_{\mathcal{A}}(q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, a, q_j)$$

and the first inequality of (c_2) follows. □

Let us build a solution ψ of (e_{12}) and (e'_2), by analysing for each variable $\mathsf{w}_{i,b,j}$ (resp. y_j), the equation (e_{12}) (resp. (e'_2)) with this variable in left-hand side.

(e_{12}).1 for $\mathsf{weight}_0(q_i, b, q_j) = \mathbb{1}$, let $\psi(\mathsf{w}_{i,b,j}) = \mathbb{1}$.

Otherwise, $\mathsf{weight}_0(q_i, b, q_j) = \mathbb{0}$:

(e_{12}).2 for $\mathsf{w}_{i,b,j} = \mathbb{0} \oplus e \oplus g \otimes \mathsf{w}_{i,b,j}$, where e does not contain $\mathsf{w}_{i,b,j}$ and g does not contain $\mathsf{w}_{i,b,j}$ nor $\oplus$, let $\psi(\mathsf{w}_{i,b,j}) = \psi(e)$.

(e_{12}).3 for $\mathsf{w}_{i,b,j} = \mathbb{0} \oplus e$ where e does not contain $\mathsf{w}_{i,b,j}$, let $\psi(\mathsf{w}_{i,b,j}) = \psi(e)$.

(e'_2).1 for $\mathsf{in}_0(q_j) = \mathbb{1}$, let $\psi(\mathsf{y}_j) = \mathbb{1}$.

Otherwise, $\mathsf{in}_0(q_j) = \mathbb{0}$:

(e'_2).2 for $\mathsf{y}_j = \mathbb{0} \oplus e \oplus g \otimes \mathsf{y}_i$, where e does not contain y_j and g does not contain y_j nor $\oplus$, let $\psi(\mathsf{y}_j) = \psi(e)$.

(e'_2).3 for $\mathsf{y}_j = e$, where e does not contain y_j, let $\psi(\mathsf{y}_j) = \psi(e)$.

Lemma 6. *ψ is a solution of (e_{12}) and (e'_2) in $\mathcal{S}$.*

Proof. First, let us observe that (e_{12}).1-3 and (e'_2).1-3 cover all the cases of equations (e_{12}), (e'_2). Indeed, $\mathsf{weight}_0(q_i, b, q_j)$ and $\mathsf{in}_0(q_j)$ can only take the values $\mathbb{0}$ and $\mathbb{1}$ and, as there are no backward transitions in the automaton, the (variable) left-hand-side of an equation can only occur at most once in the right-hand-side.

We show now that the definition of ψ is well-founded. Let us define a partial ordering $\prec$ on variables as follows:

for all $0 \leq i \leq i' \leq j' \leq j \leq m$ such that $(i', j') \neq (i, j)$, and all $a, b \in \Sigma$, $\mathsf{w}_{i',a,j'} \prec \mathsf{w}_{i,b,j}$,

for all $0 \leq i \leq j \leq m$, $0 \leq k \leq m$, and all $a \in \Sigma$, $\mathsf{w}_{i,a,j} \prec \mathsf{y}_k$.

We can observe that:

> for the equations of the form (e_{12}).3 and (e'_2).3, the variable in *lhs* is greater *wrt* $\succ$ to every variable in the *rhs*,
>
> for the equations of the form (e_{12}).2, and (e'_2).2, the variable $\mathsf{w}_{i,b,j}$ in *lhs* is $\succeq$ to every variable $\mathsf{w}_{i',a,j'}$ in the *rhs*, and $\psi(\mathsf{w}_{i,b,j})$ only depends on $\psi(\mathsf{w}_{i',a,j'})$ for $\mathsf{w}_{i,b,j} \succ \mathsf{w}_{i',a,j'}$

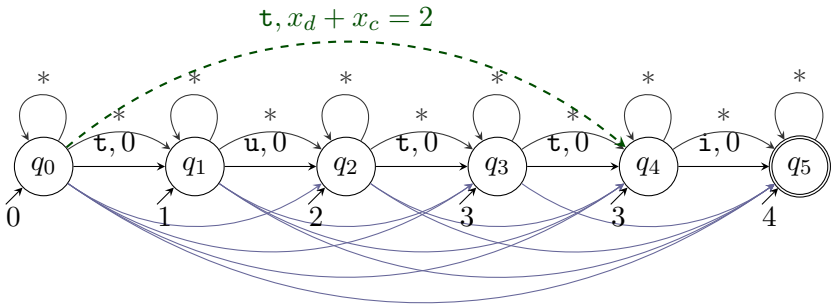

Figure 7: Generalised Levenshtein automaton $\mathcal{A}_{\texttt{tutti}}$, considering ruleset $E_0 \cup E_{3,c}$. Each state is initial. There are transitions labeled with every letter between each pair of states, with various weights. We detail the noteworthy transition $q_0 \xrightarrow{\texttt{t}} q_4$. It uses the consolidation $\texttt{ttt} \to \texttt{t}$, giving $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{ti}) = x_d + x_c = 2$. and also $\mathsf{in}(q_4) = 2 + x_d = 3$, giving $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{i}) = 3$.

It follows that ψ is well defined. And we show now that it is a solution of (e_{12}) and (e'_2).

In the case (e_{12}).1, the equation has the form $\mathsf{w}_{i,b,j} = \mathbb{1} \oplus e$, and whatever in e, $\psi(\mathbb{1} \oplus e) = \mathbb{1} \oplus \psi(e) = \mathbb{1}$ since $\mathcal{S}$ is bounded.

The case (e'_2).1 is similar.

In the case (e_{12}).2, $\psi(\mathbb{0} \oplus e \oplus g \otimes \mathsf{w}_{i,b,j}) = \psi(e) \oplus \psi(g) \otimes \psi(\mathsf{w}_{i,b,j}) = \psi(e) \otimes (\mathbb{1} \oplus \psi(g)) = \psi(e)$ since $\mathcal{S}$ is bounded.

The case (e'_2).2 is similar.

The two last cases (e_{12}).3 and (e'_2).3 are immediate. □

Completeness and Correctness of Construction Let us show now that the construction of the above automaton $\mathcal{A}_\psi$ is complete and correct, *i.e.* that it computes $D_E(s,t)$.

Theorem 7. *Assume that $\mathcal{S}$ is a commutative semiring, complete and bounded. Let E be a ruleset over $\mathcal{S}$ containing only consolidation and deletion edit operations, and let $s \in \Sigma^*$. One can build in polynomial time a weighted automaton $\mathcal{A}_s^E$ over Σ and $\mathcal{S}$ such that for all $t \in \Sigma^*$, $\mathcal{A}_s^E(t) = D_E(s,t)$.*

Let $\mathcal{A}_s^E$ over $\mathcal{S}$ and Σ be $\mathcal{A}_\psi$ constructed as above by updating the weights of the initial automaton $\mathcal{A}_s^0$ with the solution ψ of the equations (e_{12}) and (e'_2). Remember that $\mathcal{A}_s^0$ has $n+1$ states $q_0, \ldots, q_m$ where m is the length of s. Assuming that Σ is constant, the size of $\mathcal{A}_s^E$ is $O(|s|^2)$ and its construction time is in $O(|s|^4 \cdot \|E\|)$.

We decompose below the rest of the proof of Theorem 7 into two directions, respectively in Lemma 8, 9 (completeness) and Lemma 10 (correctness).

Lemma 8. *For all $t \in \Sigma^*$, and all edition sequence σ such that $s \xrightarrow[E]{\sigma} t$, it holds that $\mathcal{A}_s^E(t) \leq_{\mathcal{S}} \mathsf{weight}(\sigma)$.*

Proof. By induction on the length of the edition sequence σ. For convenience, we write $\mathcal{A}$ instead of $\mathcal{A}_s^E$.
If σ is empty, then $t = s$ and $\mathsf{weight}(\sigma) = \mathbb{1}$. There exists run a ρ of $\mathcal{A}_s^0$ over t such that $\mathsf{weight}(\rho) = \mathbb{1}$. More precisely, $\rho = \tau_1, \ldots, \tau_m$ (remember that m is the length of s), and for all $1 \leq k \leq m$, $\tau_k = \langle q_{k-1}, s_k, q_k \rangle$ and $\mathsf{weight}_{\mathcal{A}_s^0}(\tau_k) = \mathbb{1}$. By construction, ρ is also a run of $\mathcal{A}$ and for all k, $\mathsf{weight}_{\mathcal{A}}(\tau_k) = \psi(\mathsf{w}_{k-1,s_k,k}) = \mathbb{1}$ (case $(\mathsf{e}_{12}).1$ of the definition of ψ). Hence $\mathsf{weight}_{\mathcal{A}}(q_0, t, q_m) = \mathcal{A}(t) = \mathbb{1} = \mathsf{weight}(\sigma)$.

Assume that $\sigma = \sigma'\pi$ where $\pi = \langle e, p \rangle$ is a positioned operation of E, and let $x = \mathsf{weight}(e)$.
We consider first the case of a consolidation: $e = a_1 \ldots a_n \to b$ $(n \geq 0)$. The edition $s \xrightarrow[E]{\sigma} t$ can be decomposed into:

$$s \xrightarrow[E]{\sigma'} t' := w\, a_1 \ldots a_n w' \xrightarrow[E]{\pi} w\, b\, w' =: t.$$

By definition, $\mathcal{A}(t) = \bigoplus_{q,q' \in Q} \bigoplus_{\rho \in R_{\mathcal{A}}(q,t,q')} \mathsf{in}(q) \otimes \mathsf{weight}(\rho) \otimes \mathsf{out}(q')$.
For a run $\rho = \tau_1 \ldots \tau_k$ of $\mathcal{A}$ and $1 \leq i_1 \leq i_2 \leq k$, we write $\rho_{i_1..i_2}$ for the subrun $\tau_{i_1} \ldots \tau_{i_2}$.

Every run $\rho' \in R_{\mathcal{A}}(q, t', q')$ can be decomposed into $\rho' = \rho'|_w\, \rho'|_{a_1 \ldots a_n}\, \rho'|_{w'}$ where

$\rho'|_w = \rho'_{1..|w|}$ (prefix subrun of ρ' over w),

$\rho'|_{a_1 \ldots a_n} = \rho'_{|w|+1..|w|+n}$ (subrun of ρ' over $a_1 \ldots a_n$),

$\rho'|_{w'} = \rho'_{|w|+n+1..|t'|}$ (suffix subrun of ρ' over w').

It holds that $\mathsf{weight}(\rho') = \mathsf{weight}(\rho'|_w) \otimes \mathsf{weight}(\rho'|_{a_1 \ldots a_n}) \otimes \mathsf{weight}(\rho'|_{w'})$, and that $\rho = \rho'|_w \langle \mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'}) \rangle\, \rho'|_{w'}$, is a run of $R_{\mathcal{A}}(q, t, q')$, because $\langle \mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'}) \rangle$ is a transition of $\mathcal{A}$.

Similarly, every run $\rho \in R_{\mathcal{A}}(q, t, q')$ can be decomposed into

$$\rho = \rho'|_w \langle \mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'}) \rangle\, \rho'|_{w'}$$

where ρ' is a run in $R_{\mathcal{A}}(q, t', q')$ as above and $\langle \mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'}) \rangle$ is a transition of $\mathcal{A}$. It holds then that

$$\mathsf{weight}(\rho) = \mathsf{weight}(\rho'|_w) \otimes \mathsf{weight}_{\mathcal{A}}\big(\mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'})\big) \otimes \mathsf{weight}(\rho'|_{w'}).$$

Hence:

$$\begin{array}{rcl}
\mathcal{A}(t) & = & \bigoplus_{q,q' \in Q} \bigoplus_{\rho \in R_{\mathcal{A}}(q,t,q')} \mathsf{in}(q) \otimes \mathsf{weight}(\rho) \otimes \mathsf{out}(q') \\
& = & \bigoplus_{q,q' \in Q} \bigoplus_{\rho' \in R_{\mathcal{A}}(q,t',q')} \mathsf{in}(q) \otimes \mathsf{weight}(\rho'|_w) \\
& & \qquad \otimes \mathsf{weight}_{\mathcal{A}}\big(\mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'})\big) \\
& & \qquad \otimes \mathsf{weight}(\rho'|_{w'}) \otimes \mathsf{out}(q') \\
& \leq_{\mathcal{S}} & \bigoplus_{q,q' \in Q} \bigoplus_{\rho' \in R_{\mathcal{A}}(q,t',q')} \mathsf{in}(q) \otimes \mathsf{weight}(\rho'|_w) \\
& & \qquad \otimes \mathsf{weight}_{\mathcal{A}}\big(\mathsf{trg}(\rho'|_w), a_1 \ldots a_m, \mathsf{src}(\rho'|_{w'}\big) \otimes x \\
& & \qquad \otimes \mathsf{weight}(\rho'|_{w'}) \otimes \mathsf{out}(q') \\
& = & \bigoplus_{q,q' \in Q} \bigoplus_{\rho' \in R_{\mathcal{A}}(q,t',q')} \mathsf{in}(q) \otimes \mathsf{weight}(\rho') \otimes x \otimes \mathsf{out}(q') \\
& = & \mathcal{A}(t') \otimes x \\
& \leq_{\mathcal{S}} & \mathsf{weight}(\sigma') \otimes x = \mathsf{weight}(\sigma)
\end{array}$$

Note that in the second equality, we switch from a sum over runs over t ($\rho \in R_{\mathcal{A}}(q,t,q')$) to runs over t' ($\rho' \in R_{\mathcal{A}}(q,t,q')$). For this purpose, we use the following equality, which holds according to the above remarks:

$$R_{\mathcal{A}}(q,t,q') = \{\rho'|_w \langle \mathsf{trg}(\rho'|_w), b, \mathsf{src}(\rho'|_{w'})\rangle \rho'|_{w'} \;\big|\; \rho' \in R_{\mathcal{A}}(q,t',q')\}.$$

The first inequality follows from (c_1), and the last inequality holds by induction hypothesis and monotony of $\mathcal{S}$ *wrt* $\leq_{\mathcal{S}}$.

The case of a deletion: $e = a \to \varepsilon$ is treated similarly, using (c_2). □

Making the summation of the inequalities from Lemma 8 for all σ, and using the hypotheses that $\mathcal{S}$ is complete and idempotent, the completeness of $\mathcal{A}_s^E$ follows:

Lemma 9. *For all $t \in \Sigma^*$, it holds that $\mathcal{A}_s^E(t) \leq_{\mathcal{S}} D_E(s,t)$.*

The correctness of the construction of $\mathcal{A}_s^E$ follows from the next lemma.

Lemma 10. *For all $t \in \Sigma^*$, it holds that $D_E(s,t) \leq_{\mathcal{S}} \mathcal{A}_s^E(t)$.*

Proof. Every value $\mathcal{A}_s^E(t)$ returned by $\mathcal{A}_s^E$ has the form of a sum, by $\oplus$, of products, by $\otimes$, of weights of edit operations in E. Moreover, each of these products has the same form as $\mathsf{weight}(\sigma)$ for some edit sequence σ such that $s \xrightarrow{\sigma} t$. In other terms, there exists a set $P(t)$ of edit sequences σ such that $s \xrightarrow{\sigma} t$, with $\mathcal{A}_s^E(t) = \bigoplus_{\sigma \in P(t)} \mathsf{weight}(\sigma)$. This can be shown by induction on the variables, using the ordering $\prec$ introduced in the proof of Lemma 6. Thus, by idempotence for $\mathcal{S}$, it holds that:

$$\begin{array}{rcl}
\mathcal{A}_s^E(t) \oplus D_E(s,t) & = & \bigoplus_{\sigma \in P(t)} \mathsf{weight}(\sigma) \oplus \bigoplus_{s \xrightarrow{\sigma} t} \mathsf{weight}(\sigma) \\
& = & \bigoplus_{s \xrightarrow{\sigma} t} \mathsf{weight}(\sigma) \;=\; D_E(s,t)
\end{array}$$

It follows that $D_E(s,t) \leq_{\mathcal{S}} \mathcal{A}_s^E(t)$. □

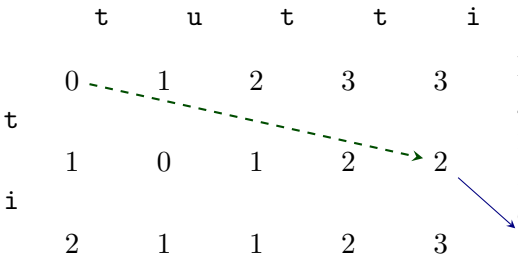

Figure 8: Computation through $\mathcal{A}^E_{\texttt{tutti}}$ of $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{ti}) = 2$. The best path in the matrix Δ corresponds to the transitions $q_0 \xrightarrow{\texttt{t}} q_4$ then $q_4 \xrightarrow{\texttt{i}} q_5$.

Note that $\mathcal{A}^E_s$ is constructed for a given s. Hence if we need to compute $D_E(s,t)$ for a fixed s and several t, we only need to compute $\mathcal{A}^E_s$ once and compute $\mathcal{A}^E_s(t)$ for each t.

Corollary 11. *Given a semiring $\mathcal{S}$ as in Theorem 7, a ruleset E over $\mathcal{S}$ containing only fragmentation and insertion edit operations, and $t \in \Sigma^*$, one can build in polynomial time a weighted automaton $\widetilde{\mathcal{A}^E_t}$ over Σ and $\mathcal{S}$ such that for all $s \in \Sigma^*$, $\widetilde{\mathcal{A}^E_t}(s) = D_E(s,t)$.*

Proof. When E contains only fragmentations and insertions, the symmetric ruleset E^{-1} contains only consolidations and deletions. The construction of the Theorem 7 can thus be applied to E^{-1} and t, giving the automaton $\widetilde{\mathcal{A}^E_t} = \mathcal{A}^{E^{-1}}_t$. Then $\widetilde{\mathcal{A}^E_t}(s) = \mathcal{A}^{E^{-1}}_t(s) = D_{E^{-1}}(t,s) = D_E(s,t)$ by commutativity of $\mathcal{S}$. □

Computation of $\mathcal{A}^E_s(t) = D_E(s,t)$ As there is no backward transition in $\mathcal{A}^E_s$, the computation of $\mathcal{A}^E_s(t) = D_E(s,t)$, given $\mathcal{A}^E_s$ can be done in a greedy way (Figure 8).

Corollary 12. *Given a semiring $\mathcal{S}$ as in Theorem 7, a ruleset E over $\mathcal{S}$ containing only consolidation and deletion edit operations, and $s,t \in \Sigma^*$, $D_E(s,t)$ can be computed in polynomial time.*

Proof. Let $\mathcal{A}^E_s$ be computed as in Theorem 7. For $0 \le i \le |s|$ and $0 \le j \le |t|$, we define $\Delta(i,j)$ recursively by

$$\begin{aligned} \Delta(i,0) &= \mathsf{in}(q_i) \\ \Delta(i,j) &= \bigoplus_{i'<i} \Delta(i', j-1) \otimes \mathsf{weight}_{\mathcal{A}^E_s}(q_{i'}, t_j, q_i) \quad 1 \le j \le |t|. \end{aligned}$$

When $\mathcal{A}^E_s$ is given, the values of Δ can be computed and stored in a table in time $O(|s|^2.|t|)$. It holds that $\Delta(|s|, j) = \mathcal{A}^E_s(t_{1..j})$. Hence, by Theorem 7, $\Delta(|s|,|t|) = D_E(s,t)$. □

Altogether, this gives a $O(|s|^4 \cdot \|E\| + |s|^2 \cdot |t|)$ algorithm to compute $D_E(s,t)$. Note that these times could be reduced by tabulating the $|E| \cdot |s|^2$ values used in the first line of (e_{12}).

4 Chaining Consolidations with Fragmentations

According to Proposition 2, one cannot generalise the automata construction of Section 3 to rulesets mixing arbitrarily consolidation and fragmentation edit operations. However, the above construction can be used to compute the optimal weight of sequences chaining operations from a ruleset E_{c}, containing consolidations and deletions, with operations from a ruleset E_{f} containing fragmentation and insertions, in that order. Given such rulesets, we define the optimal weight $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$ as follows:

$$\begin{aligned} D_{E_{\mathsf{c}},E_{\mathsf{f}}}(s,t) &= \bigoplus_{o\in\Sigma^*} D_{E_{\mathsf{c}}}(s,o)\otimes D_{E_{\mathsf{f}}}(o,t) \\ &= \bigoplus_{s\xrightarrow{\sigma}o\xrightarrow{\sigma'}t} \mathsf{weight}(\sigma)\otimes\mathsf{weight}(\sigma') \end{aligned}$$

where σ is an edit sequence of E_{c} and σ' is an edit sequence of E_{f}.

Lemma 13. *When E_{c} and E_{f} are symmetric, $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$ is a distance.*

Proof. By commutativity of $\otimes$ and idempotence of the semiring. □

Theorem 7 (and Corollary 11) can be generalised to compute $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$.

Theorem 14. *Given a commutative semiring $\mathcal{S}$, complete and bounded, E_{c} (resp. E_{f}) a ruleset over $\mathcal{S}$ containing only consolidation and deletion (resp. fragmentation and insertion) edit operations, and $s,t\in\Sigma^*$, the optimal weight $D_{E_{\mathsf{c}},E_{\mathsf{f}}}(s,t)$ can be computed in polynomial time.*

Proof. The principle is to apply the construction of Theorem 7 on the one side to s and E_{c}, giving $\mathcal{A}_s^{E_{\mathsf{c}}}$ (denoted $\mathcal{A}_s^{\mathsf{c}}$ below) and on the other side, Corollary 11 to t and E_{f}, giving $\widetilde{\mathcal{A}_t^{E_{\mathsf{f}}}} = \mathcal{A}_t^{E_{\mathsf{f}}^{-1}}$ (denoted $\widetilde{\mathcal{A}}_t^{\mathsf{f}}$ below). Altogether, $\forall o\in\Sigma^*\ \mathcal{A}_s^{\mathsf{c}}(o) = D_{E_{\mathsf{c}}}(s,o)$ and $\widetilde{\mathcal{A}}_t^{\mathsf{f}}(o) = D_{E_{\mathsf{f}}^{-1}}(t,o) = D_{E_{\mathsf{f}}}(o,t)$.

We now construct, in quadratic time, the Hadamard product $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}$ of $\mathcal{A}_s^{\mathsf{c}}$ and $\widetilde{\mathcal{A}}_t^{\mathsf{f}}$ which is a weighted automaton such that $\forall o\in\Sigma^*, \mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}(o) = \mathcal{A}_s^{\mathsf{c}}(o)\otimes \widetilde{\mathcal{A}}_t^{\mathsf{f}}(o)$ (see [3]). Let m be the element of $\mathcal{S}$ such that $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}(m)$ has minimal weight *wrt* $\leq_{\mathcal{S}}$. This element m can be computed by a Viterbi algorithm on $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}$ (see *e.g.* [6]). It holds then that $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}(m) = D_{E_{\mathsf{c}}\circ E_{\mathsf{f}}}(s,t)$. □

Note that following Proposition 2, it is not possible to iterate this procedure an arbitrary number of times. Moreover, the optimal weight of *fragmentations chained with consolidations*, in this order, is not computable.

Proposition 15. *In general, $D_{E_{\mathsf{f}},E_{\mathsf{c}}}(s,t)$ is not computable for a ruleset E_{f} containing fragmentation rules and for a ruleset E_{c} containing consolidation rules.*

Proof. We consider the rulesets E_{f} and E_{c} constructed in the proof of Proposition 2. They are like in the hypothesis of Proposition 15, and it holds that $\sharp \xrightarrow[E_{\mathsf{f}}]{*} o \xrightarrow[E_{\mathsf{c}}]{*} \natural$ for some o iff $D_{E_{\mathsf{f}},E_{\mathsf{c}}}(s,t) \neq \mathbb{0}$ iff $\mathcal{P}$ has a solution. □

Hence the result of Theorem 14, though polynomial, is close to undecidability.

The above construction can be applied to rulesets such as $E_{3,\mathsf{c}} \cup E_0$ and $E_{3,\mathsf{f}} \cup E_0$. As substitution, insertion, and deletion operations of E_0 appear in both rulesets, these operations can thus be used anywhere in $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$. Such a computation of $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$ may be used in computational music analysis. Back to the Figure 1, a transformation of the Variation 1 into the Variation 7 can be expressed as a sequence of consolidation operations followed by another sequence of fragmentation operations. One could thus evaluate the distance between two variations without knowing the underlying theme.

5 Conclusion and Perspectives

We have shown how to compute in polynomial time *extended optimal weights and edit distances*, for rulesets mixing consolidations and deletions on the one side, and fragmentations and insertions on the other side, using weighted automata constructions, in a generic semiring framework. We also shown both how to compute optimal weights of sequences of consolidations followed by fragmentations and that, in general, optimal weights of sequences of fragmentations followed by consolidations are not computable. To our knowledge, this is the first time an algorithm is proposed to compute this distance (for which dynamic programming techniques used for Levenshtein edit-distance do not apply). These results can be applied to music similarity questions that motivated the Mongeau-Sankoff algorithm as tropical semirings satisfy our assumptions of boundedness.

In [11], Mohri proposed weighted transducers to compute the Levenshtein distance between regular languages, and explained that this technique can compute more complex edit distances as long as the operations are defined by transducers. Applying it in the context of an edit distance with consolidations and fragmentations reduces to express the application of these operations with transducers – a problem that we did not try to address. Generalising the results of this paper to the computation of the extended edit distances between regular languages (or a language and a string) instead of between two strings s and t, is an open question.

The generalisation of the construction to k-closed semirings, instead of bounded ones (0-closed [10]), is another open problem.

The complexity of our procedure depends on the size of E, and this may cause scalability issues for practical applications to the analysis of written music. Indeed, in this context E may be big, as edit operations depends

on pitch or intervals and rhythms... In this case, it could makes sense to generalise our approach to a symbolic framework such as the one of [2].

References

[1] Ronald V Book and Friedrich Otto. String-rewriting systems. In *String-Rewriting Systems*, pages 35–64. Springer, 1993.

[2] Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *CAV'17*. Springer, July 2017. URL: `https://www.microsoft.com/en-us/research/publication/power-symbolic-automata-transducers-invited-tutorial/`.

[3] Manfred Droste and Werner Kuich. Semirings and formal power series. In *Handbook of Weighted Automata*, pages 3–28. Springer, 2009.

[4] S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *PNAS*, 89:10915–10919, 1992.

[5] Dieter Hofbauer and Johannes Waldmann. Deleting string rewriting systems preserve regularity. *Theoretical Computer Science*, 327(3):301–317, 2004.

[6] Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. In *In COLING*, 2008.

[7] Joseph B. Kruskal and Mark Liberman. The symmetric time-warping problem: from continuous to discrete. In *Time Warps, String Edits, and Macromolecules - The Theory and Practice of Sequence Comparison*, chapter 4. 1999.

[8] Sylvain Lombardy and Jacques Sakarovitch. The removal of weighted ε-transitions. In *International Conference on Implementation and Application of Automata*, pages 345–352. Springer, 2012.

[9] Mehryar Mohri. Generic ε-removal algorithm for weighted automata. In *Int. Conference on Implementation and Application of Automata*, pages 230–242, 2001. URL: `http://dl.acm.org/citation.cfm?id=647267.721706`.

[10] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[11] Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *Int. Journal of Foundations of Computer Science*, 14(06):957–982, 2003. `doi:10.1142/S0129054103002114`.

[12] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[13] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[14] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64:100–118, 1985.

[15] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.