Weighted Automata Computation of Edit Distances with Consolidations and Fragmentations

Mathieu Giraud, Florent Jacquemard

Weighted Automata Computation of Edit Distances with Consolidations and Fragmentations

preprint – October 31, 2018

Mathieu Giraud[a], *Florent Jacquemard*[b]

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

Abstract

We study edit distances between strings, 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 for arbitrary rulesets. We propose weighted automaton constructions to compute an edit distance taking into account both consolidations and deletions, or both fragmentations and insertions. Assuming that the operation ruleset has a constant size, these constructions are polynomial into the lengths of the involved strings. We finally show that the optimal weight of sequences made of consolidations chained with fragmentations, in that order, is computable for arbitrary rulesets, and not computable if we reverse the order of fragmentations and consolidations.

Keywords: Edit Distance, Edit Operation, Weighted Automata, Consolidation, Fragmentation, Bounded Semiring

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 [1], 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 [2] 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 humanities context, for assistance to the analysis of written music. In 1990, Mongeau and Sankoff [3] proposed to extend edit distance with *consolidation* and *fragmentation* operations to account for common transformations between melodic patterns – see Figure 1. These two operations consist basically in compressing a substring down to a single letter and expanding a single letter into multiple letters, respectively. They correspond namely to the *compression* and *expansion* operations defined in the context of dynamic time-warping algorithms for speech recognition, where they allow to compare two speech recordings with different variations in speed [4].

Edit distances between strings are usually computed with dynamic programming [1]. More generic algorithms have been proposed to compute the edit distance between two regular lan-

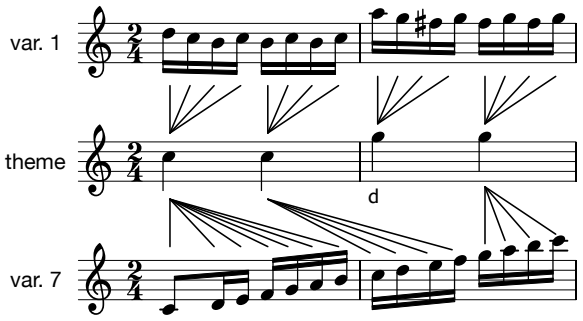

Figure 1: Theme, Variations 1 and 7 from Mozart's *"Ah vous dirai-je maman"* Variations K265. Transformations between the theme and the variations can be seen as a sequence of *fragmentation* operations, breaking a note into several ones and possibly substituting resulting notes, with also a deletion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

guages [5] with construction and composition of weighted transducers. These approaches are applied to standard Levenshtein edit-distance and are applicable to more general edit-distances defined by arbitrary sets of edit operations, provided that these operations are expressed as weighted string transducers [5]. In an unweighted context, it has been shown that some class of string rewriting rules including consolidations (as well as other kind of rules) effectively preserves regular languages [6].

In this paper, after defining edit distances with consolidation and fragmentation operations (Section 2), we show that they are not computable in general. We then propose a construction of weighted automata to compute the optimal weight of edit operations sequences including insertion and deletion 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. Definitions

In this section, we define extended edit distances between finite sequences of symbols, that can include generic *consolidation* or *fragmentation* operations (Sections 2.1 and 2.3) and where the weights are values in a semiring with some properties (Section 2.2). We also introduce the weighted automata that we shall use to compute such distances (Section 2.4).
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, uv denotes the concatenation of $u \in \Sigma^*$ and $v \in \Sigma^*$, and $|u|$ the length of u. 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 $s_{1..0} = \varepsilon$.

2.1. Edit Operations and Rulesets

The Levenshtein edit distance [1] 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 possibly 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 [7]. 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

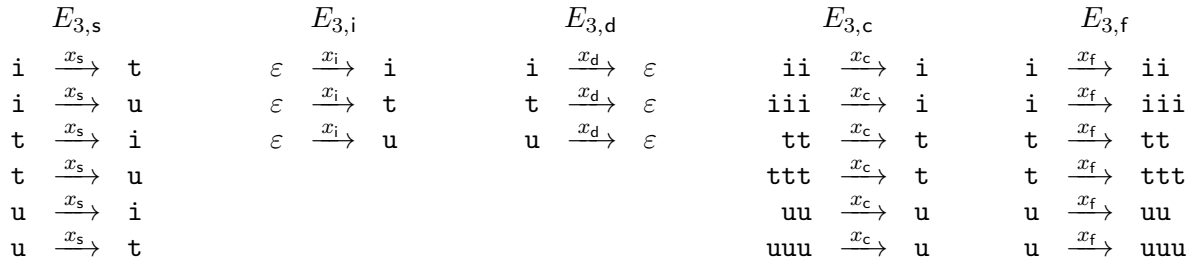

Figure 2: Rulesets over the alphabet $\Sigma_3 = \{\mathtt{i}, \mathtt{t}, \mathtt{u}\}$ for substitutions ($E_{3,\mathtt{s}}$, weight $x_\mathtt{s}$), insertions ($E_{3,\mathtt{i}}$, $x_\mathtt{i}$), deletions ($E_{3,\mathtt{d}}$, $x_\mathtt{d}$), consolidations and fragmentations up to 3 letters ($E_{3,\mathtt{c}}$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

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 $E^{-1} = \{v \to u \mid u \to v \in E\}$. The set all suffixes of a left-hand-side (*lhs*) of consolidation rule of a ruleset E is denoted $C(E)$.

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

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

In music analysis, consolidation/fragmentation rules are used to transform a note into several ones, most of the time 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 [4]. In both of these cases, it is preferable to define these operations rather than using several insertions/deletions.

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 (Example 1), 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, and there is always at least one such sequence, deleting all characters of s then inserting all characters of t.

In the case of an arbitrary E, there does not always exists a sequence transforming s into t. To capture this general situation, we consider abstract domains for weights, with two distinguished values: $\mathbb{1}$ which is a minimal distance, and $\mathbb{0}$ which corresponds to the impossibility of transforming s into t (*e.g.* infinite distance in the case of positive weight values).

More precisely, weight values are in a semiring with an operator $\otimes$, for the composition of weights in a sequence of edit operations (with neutral element $\mathbb{1}$), and an operator $\oplus$ for the selection of the optimal sequence in the definition of the edit distance (with neutral element $\mathbb{0}$). This generalisation enables the identification of the algebraic properties of the weight domain which are necessary for ensuring the correctness of our construction for the computation of edit distances.

semiring	domain	$\oplus$	$\otimes$	$\mathbb{0}$	$\mathbb{1}$	natural ordering ($\mathbb{1} \leq_{\mathcal{S}} \mathbb{0}$)
Boolean	$\{\bot, \top\}$	$\vee$	$\wedge$	$\bot$	$\top$	$\top \leq_{\mathcal{S}} \bot$
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.

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 $x \in dom(\mathcal{S})$, $x \oplus x = x$. Following the terminology of [8], when $\forall x \in dom(\mathcal{S}), \mathbb{1} \oplus x = \mathbb{1}$, the semiring $\mathcal{S}$ is is called *bounded*. Note that every bounded semiring is idempotent: by boundedness, $\mathbb{1} \oplus \mathbb{1} = \mathbb{1}$, and idempotency follows by multiplying both sides by x and distributing.

In the following applications, we need to consider infinite sums with $\oplus$. 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 and Optimal Weight

A *weighted ruleset* E is a ruleset E equipped with a *weight function*, which is a mapping $\mathsf{weight} : E \to \mathcal{S} \setminus \{\mathbb{0}\}$ 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 $u \to w \in E$,

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

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 $u \to v \in E$, $\mathsf{weight}(v \to u) = \mathsf{weight}(u \to v)$. From now on, we assume that all rulesets are weighted.

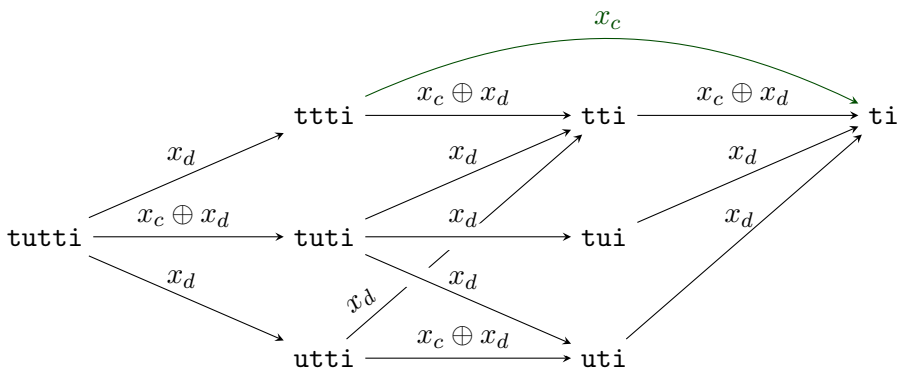

Figure 4: Subgraph of $\mathcal{G}_{E_0 \cup E_{3,c}}(\texttt{tutti})$ showing some rewriting operations from `tutti` using deletion and consolidation rules from $E_0 \cup E_{3,c}$.

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 strings $s_0, \dots, s_n \in E$. 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

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

We assume by convention that the empty sum with $\oplus$ is $\mathbb{0}$, *i.e.* when $s \neq t$ and the above sum is empty (*i.e.* $s \not\xrightarrow[E]{*} t$), 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* for 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 ($\mathbb{0} = +\infty$ and $\mathbb{1} = 0$) 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 $D_{E_0}(\texttt{tutti}, \texttt{ti}) = 3$ (three deletions), whereas $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{ti}) = 2$ (one deletion of `u` followed by one consolidation `ttt` $\to$ `t`). See Figure 4.*

2.4. Weighted Automata

In Section 3, we will present algorithms to compute D_E in some cases, using weighted automata [9].

Definition 2. *A weighted automaton $\mathcal{A}$ over the alphabet Σ and the semiring $\mathcal{S}$ is a tuple $(Q, \mathsf{in}, \mathsf{weight}, \mathsf{out})$ 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. 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 might write below $q \xrightarrow[\mathcal{A}]{a,w} q'$ to express that $w = \mathsf{weight}(q, a, q')$.
Every weighted automaton $\mathcal{A}$ defines a mapping from the strings of Σ^* into weights of $\mathcal{S}$, based on the following intermediate function $\mathsf{weight}_\mathcal{A}$ defined recursively for all $q, q' \in Q$ and all strings of Σ^*:

$$\begin{array}{rcll} \mathsf{weight}_\mathcal{A}(q, \varepsilon, q) & = & \mathbb{1} & \\ \mathsf{weight}_\mathcal{A}(q, \varepsilon, q') & = & \mathbb{0} & \text{if } q \neq q' \\ \mathsf{weight}_\mathcal{A}(q, au, q') & = & \bigoplus_{q_1 \in Q} \mathsf{weight}(q, a, q_1) \otimes \mathsf{weight}_\mathcal{A}(q_1, u, q') & \text{for } a \in \Sigma, u \in \Sigma^*. \end{array}$$

The weight associated by $\mathcal{A}$ to a string $u \in \Sigma^*$ is then defined as follows:

$$\mathcal{A}(u) = \bigoplus_{q, q' \in Q} \mathsf{in}(q) \otimes \mathsf{weight}_\mathcal{A}(q, u, q') \otimes \mathsf{out}(q').$$

Lemma 1. *For all $u, v \in \Sigma^*$, and $q, q' \in Q$, it holds that*
$\mathsf{weight}_\mathcal{A}(q, uv, q') = \bigoplus_{q_1 \in Q} \mathsf{weight}_\mathcal{A}(q, u, q_1) \otimes \mathsf{weight}_\mathcal{A}(q_1, v, q')$.

Proof. By induction on $|u|$.
The case $u = \varepsilon$ is immediate by definition of $\mathsf{weight}_\mathcal{A}(q, \varepsilon, q')$, using the facts that $\mathbb{0}$ is absorbing and $\mathbb{1}$ is neutral for $\otimes$.
If $u = au'$, then

$$\begin{array}{rcl} \mathsf{weight}_\mathcal{A}(q, uv, q') & = & \bigoplus_{q_1 \in Q} \mathsf{weight}(q, a, q_1) \otimes \mathsf{weight}_\mathcal{A}(q_1, u'v, q') \\ & = & \bigoplus_{q_1 \in Q} \mathsf{weight}(q, a, q_1) \otimes \big(\bigoplus_{q_2 \in Q} \mathsf{weight}_\mathcal{A}(q_1, u', q_2) \otimes \mathsf{weight}_\mathcal{A}(q_2, v, q')\big) \\ & = & \bigoplus_{q_1, q_2 \in Q} \mathsf{weight}(q, a, q_1) \otimes \mathsf{weight}_\mathcal{A}(q_1, u', q_2) \otimes \mathsf{weight}_\mathcal{A}(q_2, v, q') \\ & = & \bigoplus_{q_2 \in Q} \big(\bigoplus_{q_1 \in Q} \mathsf{weight}(q, a, q_1) \otimes \mathsf{weight}_\mathcal{A}(q_1, u', q_2)\big) \otimes \mathsf{weight}_\mathcal{A}(q_2, v, q')\big) \\ & = & \bigoplus_{q_2 \in Q} \mathsf{weight}_\mathcal{A}(q, u, q_2) \otimes \mathsf{weight}_\mathcal{A}(q_2, v, q'). \end{array}$$

The first and last equalities hold by definition of $\mathsf{weight}_\mathcal{A}$, the second by induction hypothesis, the two next by distributivity. □

3. Computation of 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 $\{w \mid s \xrightarrow[E]{*} w\}$, the descendants of s *wrt* E, and where each edge corresponds to one positioned operation of E and labeled by its associated weight (Figure 4). $D_E(s, t)$ is then the smallest weight of a path between s and t in $\mathcal{G}_{E,s}$ (the weight of a path π is the product with $\otimes$ of the edges forming π). However, we cannot construct $\mathcal{G}_E$ and apply a shortest-path algorithm, like in [8], to compute the optimal weight, because $\mathcal{G}_E$ is not finite in general (consider *e.g.* the case where E contains insertions). Actually, $D_E(s, t)$ is even not computable in general when E contains fragmentations and consolidations (Section 3.2). We show however that under some restrictions, we can use a weighted automaton $\mathcal{A}_s^E$ as a finite representation of $\mathcal{G}_E$ in order to compute the optimal weight (Sections 3.3 and 3.4).

3.1. The case of E_0

When E is E_0 of Example 1, then D_{E_0} can be computed in polynomial time using dynamic programming equations [10, 1, 11] providing that the triangle inequality holds. This works by tabulating $\Delta(i,j) = D_E(s_{1..i}, t_{1..j})$ with $\Delta(0,0) = \mathbb{1}$ and

$$\begin{array}{rlll} \Delta(i,j) = & \Delta(i-1,j-1) & & \text{if } s_i = t_j \quad \text{(match)} \\ \oplus & \Delta(i-1,j-1) & \otimes \mathsf{weight}(s_i \to t_j) & \text{if } s_i \neq t_j \quad \text{(substitution)} \\ \oplus & \Delta(i-1,j) & \otimes \mathsf{weight}(s_i \to \varepsilon) & \quad \text{(deletion)} \\ \oplus & \Delta(i,j-1) & \otimes \mathsf{weight}(\varepsilon \to t_j) & \quad \text{(insertion)} \end{array}$$

It holds indeed that $D_E(s,t) = \Delta(|s|,|t|)$.

Adding lines for fragmentations and consolidations to the above equation, as what was done by [3], does not always compute the optimal weight of edit operations sequences.

Example 4. *With the ruleset $E_0 \cup E_{3\mathsf{c}}$ as in Example 3 and Figure 4, we have seen that $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{ti}) = 2$. However, even after adding to the above equation a line with consolidations, we have $\Delta(5,2) = 3$. Indeed the best operation sequence includes a consolidation over a string that already underwent through a deletion, with $D_{E_0 \cup E_{3,c}}(\texttt{tutt}, \texttt{t}) = D_{E_0 \cup E_{3,c}}(\texttt{ttt}, \texttt{t}) + 1$. However* `ttt` *is not a factor of s and is not handled by Δ.*

More generally,as soon as E contains all generic insertion and deletion rules, we have always $D_E(s,t) \neq \mathbb{0}$, and $D_E(s,t)$ is computable. However, D_E is not computable in general when $E \neq E_0$ and contains fragmentations and consolidations, as shown in the next section.

3.2. Uncomputability of D_E

Proposition 1. *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 $i_1, \ldots, i_k \leq n$ (possibly with repetitions) such that $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$.

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. □

The following sections propose weighted automata constructions to correctly compute the optimal weight $D_E(s,t)$ for some rulesets E different from E_0, and including in particular fragmentations and consolidations.

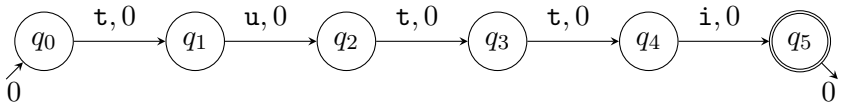

Figure 5: The weighted automaton $\mathcal{A}^0_{\texttt{tutti}}$ on the tropical semiring. All transitions that are not represented have a weight of $+\infty$.

3.3. The case of Consolidation and Deletion Rulesets

In this section we show that D_E is computable when the ruleset E contains only consolidation and deletion edit operations, as a consequence of the following theorem.

Theorem 2. *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 $s \in \Sigma^*$. On can build a weighted automaton $\mathcal{A}^E_s$ over Σ and $\mathcal{S}$ such that for all $t \in \Sigma^*$, $\mathcal{A}^E_s(t) = D_E(s,t)$. Assuming that E is of constant size, this construction is in polynomial time in $|s|$.*

We start with the proof of Theorem 2 in Sections 3.3.1 and 3.3.2 and then show how to apply it in order to to compute D_E in Section 3.3.3.

3.3.1. Automata Construction

Let $\mathcal{S}$, E, and $s \in \Sigma^*$ be like in Theorem 2. The construction of $\mathcal{A}^E_s$ starts with an automaton $\mathcal{A}^0_s$ such that $\mathcal{A}^0_s(s) = \mathbb{1}$ and then we update its transition weight function in order to make it compute the optimal weight.

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

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

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.

Example 5. *Figure 5 shows an example of $\mathcal{A}^0_s$ for $s = \texttt{tutti}$ in the tropical semiring (where $\mathbb{0} = +\infty$ and $\mathbb{1} = 0$).*

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

Lemma 2. *It holds that $\mathcal{A}^0_s(s) = \mathbb{1}$ and $\mathcal{A}^0_s(t) = \mathbb{0}$ for all $t \neq s$.*

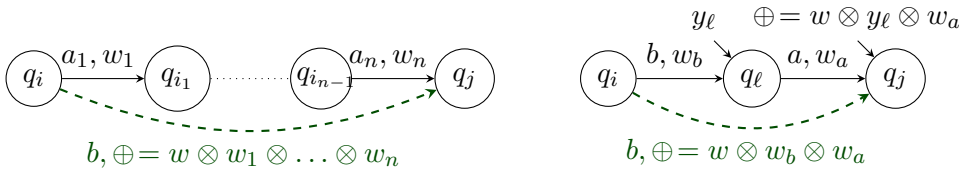

Figure 6: An illustration of the equations (3) (left, first $\bigoplus$, and bottom right, second $\bigoplus$) and (7) (top right).

Updated Automaton. The weighted automaton $\mathcal{A}_s^E$ of Theorem 2 will have the form $(Q, \mathsf{in}, \mathsf{weight}, \mathsf{out}_0)$. It is constructed from $\mathcal{A}_s^0$, leaving the state set Q unchanged while updating the entering and transition weight functions. The updated weight functions in and weight are defined by fixpoint equations over $\mathcal{S}$ and its binary operators and over the following variables.

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 $x_{i,a,j}$ representing $\mathsf{weight}(q_i, a, q_j)$, and a variable y_i representing $\mathsf{in}(q_i)$. Moreover, we consider one variable x_a for all $a \in \Sigma$, which will be used below to define the values of all $x_{i,a,i}$. We also consider one variable $z_{i,u,j}$ for all $0 \leq i < j \leq m$ and all $u \in C(E)$ (prefix of a left-hand-side of a consolidation rule in E).

Initially, every looping transitions of $\mathcal{A}_s^0$ (from one state q_i to itself) is such that $\mathsf{weight}_0(q_i, a, q_i) = \mathbb{0}$. The following equations (1) (2) define the updated weights of looping transitions, for all $b \in \Sigma$ and all $0 \leq i \leq m$.

$$x_b = \bigoplus_{\varepsilon \xrightarrow{w}_E b} w \qquad \oplus \bigoplus_{a_1 \ldots a_n \xrightarrow{w}_E b, n \geq 1} w \otimes \bigotimes_{k=1}^{n} x_{a_k} \qquad \oplus \bigoplus_{a \xrightarrow{w}_E \varepsilon} w \otimes x_a \tag{1}$$

$$x_{i,b,i} = x_b \tag{2}$$

The weights of looping transition are defined in the same way for all states, (2). The first sum in (1) treats the case of the insertion of a symbol b. In this case, while reading b, the automaton stays in state q_i, and updates the weight with the weight w of the insertion rule. The second and third sums in (1) treat respectively the case of consolidation and deletion, following principles that will be described for the next equations (3).

The updated weights of the other (non-looping) transitions are defined by the following equations (3) for all $b \in \Sigma$, and all $0 \leq i < j \leq m$,

$$x_{i,b,j} = \mathsf{weight}_0(q_i, b, q_j) \qquad \oplus \bigoplus_{a_1 \ldots a_n \xrightarrow{w}_E b, n \geq 1} w \otimes z_{i,a_1 \ldots a_n,j} \qquad \oplus \bigoplus_{a \xrightarrow{w}_E \varepsilon} \bigoplus_{i \leq \ell \leq j} w \otimes x_{i,b,\ell} \otimes x_{\ell,a,j} \tag{3}$$

And for the variables z: for all $0 \leq i \leq j \leq m$, and all $au \in C(E)$ with $a \in \Sigma$, $u \in \Sigma^*$,

$$z_{i,au,j} = \bigoplus_{i \leq \ell \leq j} x_{i,a,\ell} \otimes z_{\ell,u,j} \tag{4}$$

$$z_{i,\varepsilon,i} = \mathbb{1} \tag{5}$$

$$z_{i,\varepsilon,j} = \mathbb{0} \quad \text{if } i \neq j \tag{6}$$

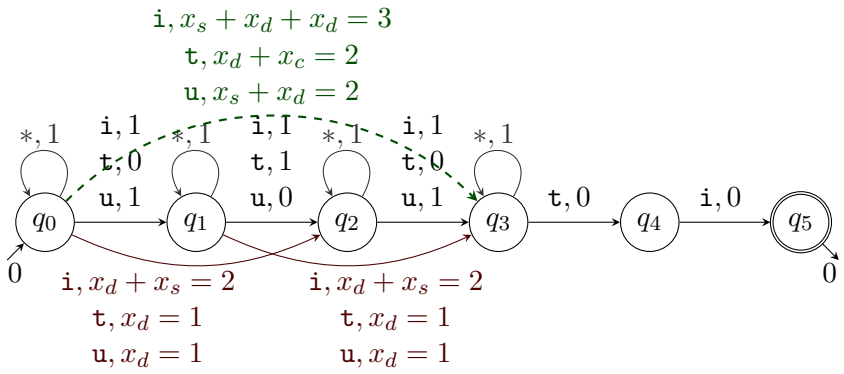

Figure 7: Automaton $\mathcal{A}^E_{\texttt{tutti}}$ on the tropical semiring after solving $\mathsf{w}_{i,a,j}$ for every $a \in \Sigma_3$ and every i and j with $0 \leq i \leq j \leq 3$, considering the ruleset $E_0 \cup E_{3,c}$. All edition rules have here a weight of 1. Solving $\mathsf{w}_{0,\texttt{t},3}$ is done through the $x_{0,\texttt{t},3}$ variable. By equation (3), we have:

$$x_{0,\texttt{t},3} = \min(+\infty,\ 1 + z_{0,\texttt{tt},3},\ 1 + z_{0,\texttt{t},3},\ \min_{a\in\Sigma_3} \min_{0\leq\ell\leq 3}(1 + x_{0,\texttt{t},\ell} + x_{\ell,a,3}))$$

By equations (4-6), $z_{0,\texttt{tt},3} = x_{0,\texttt{t},2} + x_{2,\texttt{t},3} = 1$ once all $x_{i,b,j}$ variables with $(i,j) \prec (0,3)$ have been solved. This finally gives $x_{0,\texttt{t},3} = 2$.

Figure 6 illustrates the intuitions behind the weight updates defined by (3). The first sum in (3) corresponds to the case of a consolidation $a_1 \dots a_n \xrightarrow[E]{w} b$. Intuitively, the variable $z_{i,a_1\dots a_n,j}$ is the weight for going from q_i to q_j while reading successively $a_1, \dots, a_n$, as defined in (4-6). To take into account the edition $a_1 \dots a_n \xrightarrow[E]{w} b$, we add the product of this variable and the weight w of the consolidation (into b) as a possible weight for going from q_i to q_j while reading b. Note that, unlike (2), the equations (3) do not consider insertions (condition $n \geq 1$). Indeed, since $\mathcal{A}^0_s$ contains no ε-transitions, it is needed to consider updates by insertions only for looping transition.
Finally, the equations (7) define the updated entering weights, for all $0 \leq j \leq m$.

$$y_j = \mathsf{in}_0(q_j) \oplus \bigoplus_{a \xrightarrow[E]{w} \varepsilon} \bigoplus_{\ell \leq j} w \otimes y_\ell \otimes x_{\ell,a,j} \tag{7}$$

The second sum of (3) together with the equation (7) both treat the case of a deletion. They are seen as the addition of an ε-transition $q_\ell \xrightarrow{\varepsilon} q_j$ in place of the transition labeled by a, followed by the on-the-fly suppression of this ε-transition, using a technique similar to *e.g.* [12, 13].

Example 6. *Figure 7 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).*

We show now that the above equations (1-7) have a solution in $\mathcal{S}$, and how we can effectively construct it. For this purpose, we shall use the following general Lemma 3. It uses the notion of *polynomial* over a variable set X, with coefficients in $\mathcal{S}$, which is a sum of the form: $P = a \oplus \bigoplus_{i=1}^n b_i \otimes M_i$ where $a, b_i \in \mathcal{S}$ for all $1 \leq i \leq n$ and where every M_i is a product, with $\otimes$, of variables of X (possibly with repetitions). Every element of the above sum (either a or $b_i \otimes M_i$) is called a *monomial*. The size of P is the number of monomials, and the degree of P, $deg(P)$, is the maximal number of variables inside a monomial. The set of polynomials over $\mathcal{S}$ and a set of variables X is denoted $\mathcal{S}[X]$.
For $x \in X$ and $P \in \mathcal{S}[X]$, the *loop-free form* of the equation $x = P$ is the equation $x = P'$, where P' is obtained from P by deleting every monomial that contains x. The following observation will be useful in the proof of Lemma 3.

Fact 1. *Let $\mathcal{S}$ be a commutative and bounded semiring. Let $P(x) \in \mathcal{S}[x]$ and let $x = P'$ be the loop-free form of $x = P(x)$. Then $P' = P(P')$.*

In other terms, $P' \in \mathcal{S}$ is a solution of $x = P(x)$.

Proof. By hypothesis, $P(x)$ has the following form, for some $d \geq 0$:

$$P(x) = a_0 \oplus \bigoplus_{i=1}^{d} x^i \otimes a_i$$

where $a_0, a_1, \ldots, a_d \in \mathcal{S}$ and the exponentiations x^i are built with $\otimes$. Then $P' = a_0$ and, by boundedness of $\mathcal{S}$,

$$P(a_0) = a_0 \otimes \big(\mathbb{1} \oplus \bigoplus_{i=1}^{d} a_0^{i-1} \otimes a_i\big) = a_0$$

□

Example 7. *On the tropical semiring, the equation (3) on $x_{0,\mathtt{t},3}$ in the Figure 7 has the form $x_{0,\mathtt{t},3} = \min(w, \gamma + x_{0,\mathtt{t},3})$, where $w = \min(+\infty, 1 + z_{0,\mathtt{tt},3} + ...)$ contains the weight of the best sequence of edition operations and γ gathers all terms involving again $x_{0,\mathtt{t},3}$. They correspond to cycles that will not improve the weight and are removed in the loop-free form of the equation.*

For an integer $n \geq 1$, we call *n-system* over $\mathcal{S}$ and $\{x_1, \ldots, x_n\}$ a set of n equations of the form $\big\{x_i = P_i \;\big|\; 1 \leq i \leq n, P_i \in \mathcal{S}[x_1, \ldots, x_n]\big\}$.

Lemma 3. *Let $\mathcal{S}$ be a commutative and bounded semiring. One can construct a solution for every n-system over $\mathcal{S}$.*

Proof. Let $X = \{x_1, \ldots, x_n\}$ and let U be an n-system over $\mathcal{S}$ and X. Let $U_0 = U$. For $0 < j \leq n$, assume that we have constructed $U_{j-1} = \{x_i = P_i \mid 1 \leq i \leq n\}$, and let us define the transformation of U_{j-1} into U_j as follows (elimination of x_j):

1. replace $x_j = P_j$ by its loop-free form $x_j = P'_j$.
2. for all $k \neq j$, $1 \leq k \leq n$, let Q_k be obtained from P_k by replacing every occurrence of x_j by P'_j, and normalizing the expression obtained into a polynomial form.
   Replace $x_k = P_k$ by $x_k = Q'_k$, the loop-free reduce of $x_k = Q_k$.

Fact 2. *For all $1 \leq j \leq n$, let $U_j = \{x_i = P_i \mid 1 \leq i \leq n\}$. Then for all $1 \leq i \leq n$, $P_i \in \mathcal{S}[x_{j+1}, \ldots, x_n]$.*

Proof. Straightforward induction on j, with an analysis of the above transformation steps 1 and 2. □

Thus $U_n = \{x_i = P_i \mid 1 \leq i \leq n\}$ is solved, with all P_i being values in $\mathcal{S}$.

Fact 3. *For all $1 \leq j \leq n$, every solution of U_j is a solution of U_{j-1}.*

PROOF. Let $U_{j-1} = \{x_i = P_i \mid 1 \leq i \leq n\}$. It holds that

$$U_j = \{x_1 = Q'_1,\ \ldots,\ x_{j-1} = Q'_{j-1},\ x_j = P'_j,\ x_{j+1} = Q'_{j+1},\ \ldots,\ x_n = Q'_n\}$$

where P'_j and Q'_i for $i \neq j$ are defined from U_{j-1} according to the above transformation steps 1 and 2.
Let $\sigma : \{x_1, \ldots, x_n\} \to \mathcal{S}$ be a solution of U_j. Following Fact 2, by replacing in P_j (respectively P'_j) every variable in $\{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$ by its image under σ, we obtain a polynomial $R_j(x_j) \in \mathcal{S}[x_j]$ (respectively $R'_j \in \mathcal{S}$). By Fact 1 applied to $R_j(x_j)$, it follows that $R'_j = R_j(R'_j)$. Moreover, since σ is a solution of U_j, it holds $\sigma(x_j) = R'_j$. Hence σ is a solution of $x_j = P_j$.
As Q'_i, $i \neq j$ are obtained by replacements and loop-free reduction, Fact 1 can also be applied to show that σ is a solution of the remaining equations of U_{j-1}. □

By Fact 3, the solution of U_n is a solution for $U_0 = U$. (end of proof of Lemma 3) □

Each of the n steps transform n equations. The maximum size of the polynomial in the rhs of each equation is in $O(n^d)$, where d is the maximum degree of this polynomial. Altogether, a n-system can be solved in time $O(n^{d+2})$. Note that in the worst case, $d = n \max(deg(P_i))$, where the P_i are the polynomials of the n-system to be solved.

Lemma 3 can be used to solve the equations (1-7). For efficiency, we decompose the resolution into several steps, following the dependencies between the variables.

Equations (1-2). First, we solve the $|\Sigma|$-system of equations of the form (1), using Lemma 3. This defines a solution for the variables $x_{i,b,i}$, with $1 \leq i \leq m$ and $b \in \Sigma$, according to (2). The maximum size of the polynomials in the rhs of these equations is in $O(||E||)$. In the worst case, solving this system is done in exponential time in $||E||$.

Equations (3-6). Next, we solve these equations in several steps, according to a partial ordering $\prec$ on $\{1, ..., m\} \times \{1, ..., m\}$ defined by: $(i', j') \prec (i, j)$ for all $0 \leq i \leq i' \leq j' \leq j \leq m$ such that $(i', j') \neq (i, j)$.

For all $0 \leq i \leq j \leq m$, let us denote by $X_{i,j}$ the set of variables of the form $x_{i,b,j}$ or $z_{i,u,j}$, for some $b \in \Sigma$, and $u \in C(E)$. For convenience, we denote by $x_{i,b,j} = P_{i,b,j}$ and $z_{i,u,j} = Q_{i,u,j}$ the equations in (3-6) with a left-hand side in $X_{i,j}$.

Fact 4. *For all* $0 \leq i \leq j \leq m$, $b \in \Sigma$, *and* $u \in C(E)$, $P_{i,b,j}, Q_{i,u,j} \in \mathcal{S}\big[\bigcup_{(i',j') \preceq (i,j)} X_{i',j'}\big]$.

Using the previous observation, we can solve (3-6) following the ordering $\prec$. The minimal variables *wrt* $\prec$ are in $\bigcup_{i=0}^{m} X_{i,i}$, and they have already been above solved with (2).
Then, let $0 \leq i < j \leq m$ be such that all the variables in $\bigcup_{(i',j') \prec (i,j)} X_{i',j'}$ have already been solved. Then, according to Fact 4, after replacement of the former variables by their solution in $\mathcal{S}$ in equations in (3-6) with a left-hand-side in $X_{i,j}$, we obtain a $|X_{i,j}|$-system which can be solved using Lemma 3.
The above approach solves the equations of (3-6) in $O(m^2)$ applications of Lemma 3. We have $|X_{i,j}| = O(|\Sigma| + ||E||)$, and the maximum size of the polynomials in the rhs of $X_{i,j}$ is $O(||E|| + |E|m)$. It takes $O(||E|| + |E|m)$ time to normalize the equations in $X_{i,j}$ in a polynomial form. In the worst case, solving each system is done in exponential time in $|X_{i,j}|$. Assuming that $||E||$ (and $|\Sigma|$) is constant, the $O(m^2)$ applications of Lemma 3 take a time $O(m^3)$.

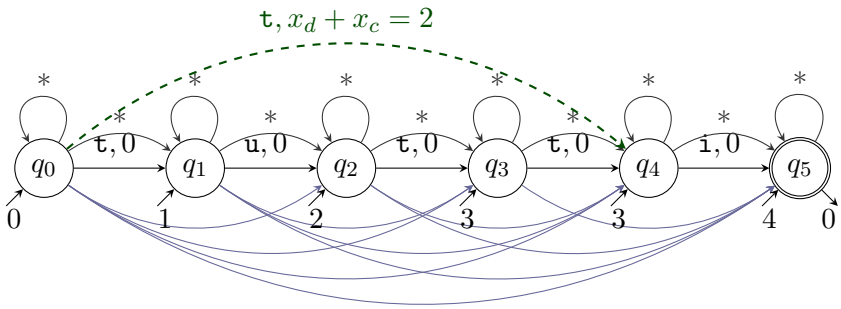

Figure 8: Generalised Levenshtein automaton $\mathcal{A}_{\texttt{tutti}}$, considering ruleset $E_0 \cup E_{3,c}$. Each state is initial. There are transitions labeled with every letter between each pair of states, with various weights. We detail the noteworthy transition $q_0 \xrightarrow{\texttt{t}} q_4$. 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 $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{i}) = 3$.

Equation (7). Finally, after a replacement of all variables in $X_{i,j}$ by their solution in $\mathcal{S}$, we can solve in $O(m)$ applications of the Fact 1 the equations (7) on y_j, starting from $j = 0$ until $j = m$. Assuming that $||E||$ is constant, these applications take a time $O(m^2)$.

Let σ be a solution of the equations (1-7), computed as above. We define the the entering and transition weight functions of the automaton $\mathcal{A}_s^E$ by:

for all $1 \leq i \leq j \leq m$ and all $a \in \Sigma$, $\mathsf{weight}(q_i, a, q_j) = \sigma(x_{i,a,j})$,

for all $1 \leq j < i \leq m$ and all $a \in \Sigma$, $\mathsf{weight}(q_i, a, q_j) = \mathsf{weight}_0(q_i, a, q_j) = \mathbb{0}$,

for all $1 \leq i \leq m$, $\mathsf{in}(q_i) = \sigma(y_i)$.

To summarize, the weighted automaton $\mathcal{A}_s^E$ has $m+1$ states $q_0, \ldots, q_m$ where m is the length of s. Assuming that Σ and E are constant, the size of $\mathcal{A}_s^E$ is in $O(|s|^2)$ and its construction time is in $O(|s|^3)$.

3.3.2. Completeness and Correctness

We shall now terminate the proof of Theorem 2 by showing that the weighted automaton $\mathcal{A}_s^E$ over $\mathcal{S}$ and Σ constructed in the above section 3.3.1 is such that for all $t \in \Sigma^*$, $\mathcal{A}_s^E(t) = D_E(s,t)$. The proof is decomposed into two directions, respectively in Lemmata 4, 5 (completeness) and Lemma 6 (correctness).

In this section, we write $\mathcal{A}$ for $\mathcal{A}_s^E$ in order to simplify notations. Let σ be the solution of (1-7) used in section 3.3.1 for the construction of $\mathcal{A}$. The following fact is an immediate consequence of (4-6) and the definition of $\mathsf{weight}_{\mathcal{A}}$.

Fact 5. *For all* $0 \leq i \leq j \leq m$, $b \in \Sigma$, $u \in C(E)$,
$\mathsf{weight}_{\mathcal{A}}(q_i, b, q_j) = \sigma(z_{i,b,j}) = \sigma(x_{i,b,j})$ *and* $\mathsf{weight}_{\mathcal{A}}(q_i, u, q_j) = \sigma(z_{i,u,j})$.

Fact 6. *For all* consolidation $a_1 \ldots a_n \xrightarrow{w} b \in E$ *with* $a_1, \ldots, a_n, b \in \Sigma$ *and* $n \geq 1$, *for all states* $q_i, q_j \in Q$, *with* $0 \leq i \leq j \leq m$, *it holds that*

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

Proof. If $i < j$, $w \otimes z_{i,a_1 \ldots a_n,j}$ is a monomial of the rhs of (3) where the lhs is $x_{i,b,j}$, and by idempotency of $\mathcal{S}$, it holds that $x_{i,b,j} \oplus w \otimes z_{i,a_1 \ldots a_n,j} = x_{i,b,j}$. Hence $\sigma(x_{i,b,j}) \leq_{\mathcal{S}} w \otimes \sigma(z_{i,a_1 \ldots a_n,j})$. Similarly if $i = j$, then $w \otimes \bigotimes_{k=1}^{n} x_{a_k}$ is a monomial of (1) and $\sigma(x_{i,b,i}) = \sigma(x_b) \leq_{\mathcal{S}} w \otimes \bigotimes_{k=1}^{n} \sigma(x_{i,a_k,i})$, where $\sigma(x_{i,b,i}) = \sigma(x_b)$ and $\sigma(x_{i,a_k,i}) = \sigma(x_{a_k})$ follow from (1).
In both cases, the inequality of Fact 6 follows, by Fact 5. □

Fact 7. *For all* insertion $\varepsilon \xrightarrow{w} a \in E$ *with* $a \in \Sigma$*, for all* $q_i \in Q$*, with* $0 \leq i \leq m$*, it holds that* $\mathsf{weight}_{\mathcal{A}}(q_i, a, q_i) \leq_{\mathcal{S}} w$.

Proof. The weight w is a monomial of (1) (first sum), and hence $\sigma(x_{i,a,i}) = \sigma(x_a) \leq_{\mathcal{S}} w$. As in the proof of Fact 6, we conclude using Fact 5. □

Fact 8. *For all* deletion $a \xrightarrow{w} \varepsilon \in E$ *with* $a \in \Sigma$*, for all states* $q_\ell, q_j \in Q$ *with* $0 \leq \ell \leq j \leq m$*, it holds that*

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

and for any state $q_i \in Q$ *with* $0 \leq i \leq \ell$*, it holds that*

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

Proof. The principle is the same than in the proof of Fact 6, considering the form of the equations (3) and (7). □

Lemma 4. *For all* $t \in \Sigma^*$*, and all edition sequence* σ *such that* $s \xrightarrow[E]{\sigma} t$*, it holds that* $\mathcal{A}^E_s(t) \leq_{\mathcal{S}} \mathsf{weight}(\sigma)$.

Proof. By induction on the length of the edition sequence σ – again, we write $\mathcal{A}$ instead of $\mathcal{A}^E_s$.

If σ is empty, then $t = s$ and $\mathsf{weight}(\sigma) = \mathbb{1}$. By construction of $\mathcal{A}^0_s$, $\mathcal{A}^0_s(t) = \mathcal{A}^0_s(s) = \mathsf{weight}_{\mathcal{A}^0_s}(q_0, s, q_m) = \bigotimes_{i=1}^{m} \mathsf{weight}_{\mathcal{A}^0_s}(q_{i-1}, s_i, q_i) = \mathbb{1}$. That still holds for $\mathcal{A}$, by construction and boundedness of $\mathcal{S}$.

Assume that $\sigma = \sigma'\pi$ where $\pi = \langle e, p\rangle$ is a positioned operation of E, and let $w = \mathsf{weight}(e)$.

Consolidations. We consider first the case of a consolidation: $e = a_1 \dots a_n \xrightarrow{w} b$ $(n \geq 1)$. In this case, the edition $s \xrightarrow[E]{\sigma} t$ can be decomposed into:

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

For $q_i, q_j \in Q$, with $0 \leq i \leq j \leq m$, by Lemma 1, and using the fact that $\mathcal{A}$ has no backward transition, it holds that

$$\begin{aligned}
\mathsf{weight}_{\mathcal{A}}(q_i, t, q_j) &= \bigoplus_{i \leq k \leq j} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, bu', q_j) \\
&= \bigoplus_{i \leq k \leq j} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \Big(\bigoplus_{k \leq \ell \leq j} \mathsf{weight}_{\mathcal{A}}(q_k, b, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j)\Big) \\
&= \bigoplus_{i \leq k \leq \ell \leq j} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, b, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j).
\end{aligned}$$

Moreover, using Lemma 1 twice,

$$\mathsf{weight}_{\mathcal{A}}(q_i, t', q_j) = \bigoplus_{i \leq k \leq \ell \leq j} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, a_1 \dots a_n, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j).$$

Therefore, by Fact 6, and using monotonicity of $\mathcal{S}$ and distributivity of $\otimes$,

$$\mathsf{weight}_{\mathcal{A}}(q_i, t, q_j) \leq_{\mathcal{S}} w \otimes \mathsf{weight}_{\mathcal{A}}(q_i, t', q_j). \tag{8}$$

This inequality still holds for $q_i, q_j \in Q$ with $0 \leq j < i \leq m$. In this case, $\mathsf{weight}_{\mathcal{A}}(q_i, t, q_j) = \mathsf{weight}_{\mathcal{A}}(q_i, t', q_j) = \mathbb{0}$ since we do not update the weight of backward transitions in the construction of $\mathcal{A}$ from $\mathcal{A}^0_s$.

Multiplying each side of (8) by $\mathsf{in}(q_i)$ and $\mathsf{out}(q_j)$ and summing up the inequalities, by monotonicity of $\mathcal{S}$ and distributivity of $\otimes$, it follows that $\mathcal{A}(t) \leq_{\mathcal{S}} w \otimes \mathcal{A}(t')$. Hence, by induction hypothesis and monotony of $\mathcal{S}$:

$$\mathcal{A}(t) \leq_{\mathcal{S}} w \otimes \mathsf{weight}(\sigma') = \mathsf{weight}(\sigma).$$

Deletions. Let us now consider the case of a deletion: $e = a \xrightarrow{w} \varepsilon$. In this case, $s \xrightarrow[E]{\sigma} t$ can be decomposed into:

$$s \xrightarrow[E]{\sigma'} t' := u\,a\,u' \xrightarrow[E]{\pi} u\,u' =: t.$$

If $u = \varepsilon$, then

$$\begin{aligned}
\mathcal{A}(t) &= \bigoplus_{q_k,q_j \in Q} \mathsf{in}(q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, u', q_j) \otimes \mathsf{out}(q_j) \\
\mathcal{A}(t') &= \bigoplus_{q_i,q_j \in Q} \mathsf{in}(q_i) \otimes \mathsf{weight}_{\mathcal{A}}(q_i, au', q_j) \otimes \mathsf{out}(q_j) \\
&= \bigoplus_{q_i,q_k,q_j \in Q} \mathsf{in}(q_i) \otimes \mathsf{weight}_{\mathcal{A}}(q_i, a, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, u', q_j) \otimes \mathsf{out}(q_j).
\end{aligned}$$

We can actually restrict the above sums to the cases $i \leq k \leq j$ since backward transitions have weight $\mathbb{0}$ in $\mathcal{A}$. From the first part of Fact 8, $\mathsf{in}(q_k) \leq_{\mathcal{S}} w \otimes \mathsf{in}(q_i) \otimes \mathsf{weight}_{\mathcal{A}}(q_i, a, q_k)$, and using monotonicity of $\mathcal{S}$ and distributivity of $\otimes$, it follows that $\mathcal{A}(t) \leq_{\mathcal{S}} w \otimes \mathcal{A}(t')$, and we conclude using the induction hypothesis as in the previous case.

If $u \neq \varepsilon$, for $q_i, q_j \in Q$, again by Lemma 1, it holds that on the one hand,

$$\mathsf{weight}_{\mathcal{A}}(q_i, t, q_j) = \bigoplus_{q_\ell \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j)$$

and on the other hand,

$$\begin{aligned}
\mathsf{weight}_{\mathcal{A}}(q_i, t', q_j) &= \bigoplus_{q_k \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, au', q_j) \\
&= \bigoplus_{q_k \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \big(\bigoplus_{q_\ell \in Q} \mathsf{weight}_{\mathcal{A}}(q_k, a, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j) \big) \\
&= \bigoplus_{q_k,q_\ell \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, a, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j).
\end{aligned}$$

In order to compare these two expressions, we use the following fact.

Fact 9. *For all $a \in \Sigma$, $u \in \Sigma^* \setminus \{\varepsilon\}$, $q_i, q_\ell \in Q$, it holds that*

$$\mathsf{weight}_{\mathcal{A}}(q_i, u, q_\ell) \leq_{\mathcal{S}} \bigoplus_{q_k \in Q} w \otimes \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, a, q_\ell).$$

Proof. By induction on the length of u, using Fact 8. □

From Fact 9 and the above identities, it follows that

$$\mathsf{weight}_{\mathcal{A}}(q_i, t, q_j) \leq_{\mathcal{S}} w \otimes \mathsf{weight}_{\mathcal{A}}(q_i, t', q_j).$$

for all $q_i, q_j \in Q$, and we conclude as in the above case of consolidation.

Insertions. It remains to consider the case of an insertion: $e = \varepsilon \xrightarrow{w} a$. In this case, $s \xrightarrow[E]{\sigma} t$ can be decomposed into:

$$s \xrightarrow[E]{\sigma'} t' := u\,u' \xrightarrow[E]{\pi} u\,a\,u' =: t.$$

We have:

$$\begin{aligned}
\mathsf{weight}_{\mathcal{A}}(q_i, t, q_j) &= \bigoplus_{q_k \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, au', q_j) \\
&= \bigoplus_{q_k, q_\ell \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, a, q_\ell) \otimes \mathsf{weight}_{\mathcal{A}}(q_\ell, u', q_j) \\
&\leq_{\mathcal{S}} \bigoplus_{q_k \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, a, q_k) \otimes \mathsf{weight}_{\mathcal{A}}(q_k, u', q_j) \\
&\leq_{\mathcal{S}} \bigoplus_{q_k \in Q} \mathsf{weight}_{\mathcal{A}}(q_i, u, q_k) \otimes w \otimes \mathsf{weight}_{\mathcal{A}}(q_k, u', q_j) \\
&= w \otimes \mathsf{weight}_{\mathcal{A}}(q_i, t', q_j).
\end{aligned}$$

The second inequality is by Fact 7, and the last line by Lemma 1. We can conclude as in the other cases. □

Making the summation of the inequalities from Lemma 4 for all σ, and using the hypotheses that $\mathcal{S}$ is complete and idempotent, we have the completeness of $\mathcal{A}_s^E$:

Lemma 5. *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 6. *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 $\mathbb{0}/\mathbb{1}$ values for in_0, weight_0, and out_0, as well as of weights of edit operations in E. Once the $\mathbb{0}/\mathbb{1}$ values have been removed, each of the non-$\mathbb{1}$ products has the same form as $\mathsf{weight}(\sigma)$ for some edit sequence σ such that $s \xrightarrow{\sigma} t$. This can be shown by induction using the ordering $\prec$ defined in Section 3.3.1. In other terms, there exists a set $P(t)$ of edit sequences σ such that $s \xrightarrow{\sigma} t$, with $\mathcal{A}_s^E(t) = \bigoplus_{\sigma \in P(t)} \mathsf{weight}(\sigma)$. Thus, by idempotency for $\mathcal{S}$, it holds that:

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

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

3.3.3. Effective Computation of D_E

Now that we have constructed $\mathcal{A}_s^E$ as in Theorem 2, we can use it to compute $D_E(s,t)$ for a given $t \in \Sigma^*$. As there is no backward transition in $\mathcal{A}_s^E$, the computation of $\mathcal{A}_s^E(t) = D_E(s,t)$, given $\mathcal{A}_s^E$ can be done in a greedy way (Figure 9).

Corollary 3. *Given a semiring $\mathcal{S}$ as in Theorem 2, a ruleset E over $\mathcal{S}$ containing only consolidation and deletion edit operations, and $s, t \in \Sigma^*$, $D_E(s,t)$ is computable.*

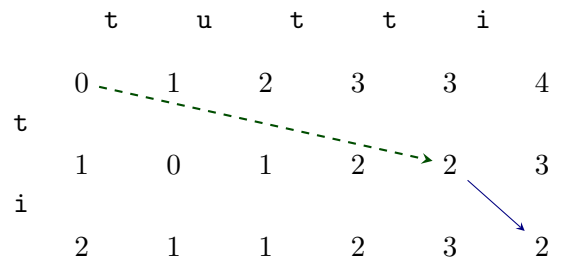

Figure 9: Computation through $\mathcal{A}^E_{\texttt{tutti}}$ of $D_{E_0 \cup E_{3,c}}(\texttt{tutti}, \texttt{ti}) = 2$. The best path in the matrix Δ corresponds to the transitions $q_0 \xrightarrow{\texttt{t}} q_4$ then $q_4 \xrightarrow{\texttt{i}} q_5$.

PROOF. Let $\mathcal{A}^E_s$ be computed as in Theorem 2. For $0 \leq i \leq |s|$ and $0 \leq j \leq |t|$, we define $\Delta(i,j)$ recursively by

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

Once $\mathcal{A}^E_s$ has been constructed, the values of Δ can be computed and stored in a table in time $O(|s|^2.|t|)$. It holds that $\Delta(i,j) = \bigoplus_{i' \leq i} \mathsf{in}(q_{i'}) \otimes \mathsf{weight}_{\mathcal{A}}(q_{i'}, t_{1..j}, q_i)$. Hence, by Theorem 2, $\Delta(|s|,|t|) = \mathcal{A}^E_s(t) = D_E(s,t)$. □

As in the Theorem 2, the algorithm computing $D_E(s,t)$ takes a time exponential in $||E||$. Assuming that E is constant, the whole computation of $D_E(s,t)$ runs in $O(|s|^3 + |s|^2 \cdot |t|)$.

Note that once $\mathcal{A}^E_s$ has been constructed for a given s, it can be reused to compute $D_E(s,t)$ for several t, using in $O(|s|^2 \cdot |t|)$ time the tabulation algorithm in the proof above.

3.4. The case of Fragmentation and Insertion Rulesets

The construction of Theorem 2 can also be reused for computing D_E for a symmetric case of ruleset E.

Corollary 4. *Given a semiring $\mathcal{S}$ as in Theorem 2, a ruleset E over $\mathcal{S}$ containing only fragmentation and insertion edit operations, and $t \in \Sigma^*$, one can build a weighted automaton $\mathcal{B}^E_t$ over Σ and $\mathcal{S}$ such that for all $s \in \Sigma^*$, $\mathcal{B}^E_t(s) = D_E(s,t)$.*

PROOF. When E contains only fragmentations and insertions, the symmetric ruleset E^{-1} contains only consolidations and deletions. The construction of the Theorem 2 can thus be applied to E^{-1} and t, giving the automaton $\mathcal{B}^E_t = \mathcal{A}^{E^{-1}}_t$. Then $\mathcal{B}^E_t(s) = \mathcal{A}^{E^{-1}}_t(s) = D_{E^{-1}}(t,s) = D_E(s,t)$ by commutativity of $\mathcal{S}$. □

Using Corollary 4 and the technique used in the proof of Corollary 3, we obtain the following corollary.

Corollary 5. *Given a semiring $\mathcal{S}$ as in Theorem 2, a ruleset E over $\mathcal{S}$ containing only fragmentation and insertion edit operations, and $s, t \in \Sigma^*$, $D_E(s,t)$ is computable.*

4. Chaining Consolidations with Fragmentations

According to Proposition 1, 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 7. *When E_{c} and E_{f} are symmetric, $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$ is a distance.*

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

Theorem 2 (and Corollary 4) can be generalised to compute $D_{E_{\mathsf{c}},E_{\mathsf{f}}}$.

Theorem 6. *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 $D_{E_{\mathsf{c}},E_{\mathsf{f}}}(s,t)$ is computable.*

Proof. The principle is to apply the construction of Theorem 2 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 4 to t and E_{f}, giving $\mathcal{B}_t^{E_{\mathsf{f}}}=\mathcal{A}_t^{E_{\mathsf{f}}^{-1}}$ (denoted $\mathcal{B}_t^{\mathsf{f}}$ below). Altogether, for every $o\in\Sigma^*$, we have $\mathcal{A}_s^{\mathsf{c}}(o)=D_{E_{\mathsf{c}}}(s,o)$ and $\mathcal{B}_t^{\mathsf{f}}(o)=D_{E_{\mathsf{f}}^{-1}}(t,o)=D_{E_{\mathsf{f}}}(o,t)$.

We now construct, in quadratic time, the Hadamard product $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}$ of $\mathcal{A}_s^{\mathsf{c}}$ and $\mathcal{B}_t^{\mathsf{f}}$ which is a weighted automaton such that $\forall o\in\Sigma^*, \mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}(o)=\mathcal{A}_s^{\mathsf{c}}(o)\otimes\mathcal{B}_t^{\mathsf{f}}(o)$ (see [9]).
Let $o'\in\Sigma^*$ such that $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}(o')$ has minimal weight *wrt* $\leq_{\mathcal{S}}$. This o' can be computed by a Viterbi algorithm on $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}$ (see *e.g.* [14]). It holds then that $\mathcal{A}_{s,t}^{\mathsf{c},\mathsf{f}}(o')=D_{E_{\mathsf{c}}\circ E_{\mathsf{f}}}(s,t)$. □

Note that following Proposition 1, 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 7. *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 1. They are like in the hypothesis of Proposition 7, 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 6 is close to undecidability. This 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 here the distance between two variations without knowing the underlying theme.

5. Conclusion and Perspectives

We have shown how to compute *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, even if the time complexity in E of the proposed constructions may cause scalability issues.

In [5], 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 closed semirings [15], instead of bounded ones, is another open problem.

Acknowledgements. We thank anonymous reviewers for their comments on an earlier version of this manuscript.

References

[1] R. A. Wagner, M. J. Fischer, The string-to-string correction problem, Journal of the ACM (JACM) 21 (1) (1974) 168–173.

[2] S. Henikoff, J. Henikoff, Amino acid substitution matrices from protein blocks, PNAS 89 (1992) 10915–10919.

[3] M. Mongeau, D. Sankoff, Comparison of musical sequences, Computers and the Humanities 24 (3) (1990) 161–175.

[4] J. B. Kruskal, M. Liberman, The symmetric time-warping problem: from continuous to discrete, in: Time Warps, String Edits, and Macromolecules – The Theory and Practice of Sequence Comparison, 1999, Ch. 4.

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

[6] D. Hofbauer, J. Waldmann, Deleting string rewriting systems preserve regularity, Theoretical Computer Science 327 (3) (2004) 301–317.

[7] R. V. Book, F. Otto, String-rewriting systems, in: String-Rewriting Systems, Springer, 1993, pp. 35–64.

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

[9] M. Droste, W. Kuich, Semirings and formal power series, in: Handbook of Weighted Automata, Springer, 2009, pp. 3–28.

[10] S. Needleman, C. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of Molecular Biology 48 (3) (1970) 443–453.

[11] E. Ukkonen, Algorithms for approximate string matching, Information and Control 64 (1985) 100–118.

[12] M. Mohri, Generic ε-removal algorithm for weighted automata, in: Int. Conference on Implementation and Application of Automata (CIAA), 2001, pp. 230–242.

[13] S. Lombardy, J. Sakarovitch, The removal of weighted ε-transitions, in: Int. Conference on Implementation and Application of Automata (CIAA), Springer, 2012, pp. 345–352.

[14] L. Huang, Advanced dynamic programming in semiring and hypergraph frameworks, in: Int. Committee on Computational Linguistics Conference (COLING), 2008.

[15] S. Dolan, Fun with semirings: A functional pearl on the abuse of linear algebra, in: Int. Conference on Functional Programming (ICFP), 2013.