Top-Down Tree Edit-Distance of Regular Tree Languages[⋆]

Sang-Ki Ko[1], Yo-Sub Han[1], and Kai Salomaa[2]

[1] Department of Computer Science, Yonsei University
50, Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea
{narame7,emmous}@cs.yonsei.ac.kr
[2] School of Computing, Queen's University
Kingston, Ontario K7L 3N6, Canada
ksalomaa@cs.queensu.ca

Abstract. We study the edit-distance of regular tree languages. The edit-distance is a metric for measuring the similarity or dissimilarity between two objects, and a regular tree language is a set of trees accepted by a finite-state tree automaton or described by a regular tree grammar. Given two regular tree languages L and R, we define the edit-distance $d(L, R)$ between L and R to be the minimum edit-distance between a tree $t_1 \in L$ and $t_2 \in R$, respectively. Based on tree automata for L and R, we present a polynomial algorithm that computes $d(L, R)$. We also suggest how to use the edit-distance between two tree languages for identifying a special common string between two context-free grammars.

Keywords: tree edit-distance, regular tree languages, tree automata, dynamic programming.

1 Introduction

It is an important problem to measure the similarity or dissimilarity between data in many applications [14,22,24]. For example, there are several similarity measures between two strings [7,11,21] and one of the most well-known measures is is the Levenshtein distance [11], which is often called the *edit-distance* in the literature. The edit-distance problem is, then, to compute the shortest distance between inputs. Researchers extended the edit-distance problem between strings into the edit-distance problem between a string and a language, or between two languages [3,8,9,12,13,19,20]. Another extension of the string edit-distance problem is the tree edit-distance problem [5,10,16,17,23]. The *tree edit-distance* problem is to find the minimum number of edit operations required to transform one tree into the other. The tree edit-distance plays an important role for calculating the similarity between structural data such as XML documents [18].

[⋆] Ko and Han were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562), and Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

Consider a tree t of size m (namely, there are m nodes in t) and let m_h and m_l be the height and the number of leaves of t. Tai [17] considered the problem of computing the tree edit-distance as a generalization of the string edit-distance problem and designed an $\mathcal{O}(m_l^2 n_l^2 mn)$ algorithm, where m and n are the sizes of input trees. Later, Shasha and Zhang [23] improved Tai's algorithm and presented an $\mathcal{O}(mn \cdot \min\{m_h, m_l\} \cdot \min\{n_h, n_l\})$ algorithm, which runs in $\mathcal{O}(m^2n^2)$ time in the worst-case. Klein [10] further improved this algorithm and obtained an $\mathcal{O}(m^2 n \log n)$ algorithm, where $n \geq m$. Recently, Demaine et al. [5] suggested an $\mathcal{O}(m^2n(1 + \log \frac{n}{m}))$ time algorithm, for $n \geq m$, using an optimal decomposition strategy. Note that all these algorithms allows both insertion or deletion of internal nodes in a tree.

Selkow [16] considered the tree edit-distance model that requires insertion and deletion to be allowed only at leaf nodes and called this problem the *top-down tree edit-distance problem.* Then, he designed an $\mathcal{O}(mn)$ algorithm for the problem. Researchers successfully applied the top-down tree edit-distance to several applications [2,14,15]. For instance, Nierman and Jagadish [14] considered several tree edit-distance definitions for clustering XML documents and demonstrated that top-down tree edit-distance guarantees less mis-clusterings than the general tree edit-distance and, thus, is a better clustering scheme.

We examine the top-down tree edit-distance of two regular tree languages accepted by tree automata. There are many results on the problem of computing the distance between languages [1,3,8,9,12]. A regular tree language is a set of trees, and is specified by either a regular tree grammar or a finite-state tree automaton. We propose an $\mathcal{O}(m^2n^2)$ algorithm for computing the edit-distance between two regular tree languages of k-bounded trees and an $\mathcal{O}(m^2n^2 \log mn)$ algorithm for the edit-distance between two regular tree languages of unbounded trees, where m and n are sizes of two input tree automata.

In Section 2, we give basic definitions and notations. Then, we introduce the tree edit-distance problem in Section 3. We propose an algorithm for the edit-distance between two regular tree languages of k-bounded trees in Section 4. We also consider the unranked case in Section 5. Then, we show that our result can be applied to the problem of measuring the similarity between two context-free string languages in Section 6 and conclude the paper in Section 7.

2 Preliminaries

A ranked alphabet Σ is a pair of a finite set of characters and a function $r : \sigma \to \mathbb{N} \cup \{0\}$. We denote the set of elements of rank m by $\Sigma_m \subseteq \Sigma$, where $m \geq 0$. The set F_Σ consists of Σ-labeled trees, where a node labeled by $\sigma \in \Sigma_m$ for $m \geq 0$ has exactly m children. We denote the set of trees over Σ by F_Σ, which is the smallest set S satisfying the following condition: if $m \geq 0, \sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in S$, then $\sigma(t_1, \ldots, t_m) \in S$. In an unranked tree each node has a finite number of children but the label of a node does not determine the number of children and there is no apriori upper bound on the number of children. Unranked trees can be defined as above by replacing the condition "$\sigma \in \Sigma_m$" by "$\sigma \in \Sigma$".

A *finite-state automaton* (FA) A is specified by a tuple $A = (\Sigma, Q, F, \delta)$, where Σ is an alphabet, Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ is a transition function. Given an FA $A = (Q, \Sigma, F, \delta)$, we define the size $|A|$ of A to be $|Q| + |\mathrm{dom}(\delta)|$. Note that an FA accepts a regular language.

A *nondeterministic ranked tree automaton* (NTA) A is specified by a tuple (Q, Σ, F, δ), where Q is a finite set of states, Σ is a ranked alphabet, $F \subseteq Q$ is a set of final states, and δ associates to each $\sigma \in \Sigma_m$ a mapping $\sigma_\delta : Q^m \to 2^Q, m \geq 0$. For each tree $t = \sigma(t_1, \ldots, t_m) \in F_\Sigma$, we define inductively the set $t_\delta \subseteq Q$ by setting $q \in t_g$ if and only if there exist $q_i \in (t_i)_\delta$, for $1 \leq i \leq m$, such that $q \in \sigma_\delta(q_1, \ldots, q_m)$. Intuitively, t_δ consists of the states of Q that A may reach by reading the tree t. Thus, the tree language accepted by A is defined as follows: $L(A) = \{t \in F_\Sigma \mid t_\delta \cap Q_f \neq \emptyset\}$. We define the size $|A|$ of a ranked TA A to be $|Q| + \sum_{\sigma_\delta(q_1,\ldots,q_m) \to q}(r(\sigma) + 1)$. The automaton A is a *deterministic ranked tree automaton* (DTA) if, for each $\sigma \in \Sigma_m$, where $m \geq 0$, σ_δ is a partial function $Q^m \to Q$. The nondeterministic (bottom-up or top-down) and deterministic bottom-up tree automata accept the family of *regular tree languages of ranked trees.*

A *nondeterministic unranked tree automaton* is specified by a tuple $A = (\Sigma, Q, F, \delta)$, where Σ is an alphabet, Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ is a transition relation. For each $q \in Q$ and $\sigma \in \Sigma$, we define $\delta(q, \sigma)$ to be the horizontal language associated with q and σ. We denote an FA for the horizontal language $\delta(q, a)$ of A by $H^A_{q,\sigma}$. Then, according to the transition relation δ, each $\sigma \in \Sigma$ defines a partial function $\sigma_\delta : Q^* \to Q$, where, for $w \in Q^*$, $q \in Q$, $q \in \sigma_\delta(w)$ if $w \in H^A_{q,\sigma}$. The tree language accepted by A is defined as follows: $L(A) = \{t \in T_\Sigma \mid t \xrightarrow{*} q_f \in F\}$. We define the size $|A|$ of an unranked TA A to be $|Q| + \sum_{q \in Q, \sigma \in \Sigma}(|H^A_{q,\sigma}| + 1)$. Naturally a ranked tree automaton is a special case of an unranked tree automaton, where for $\sigma \in \Sigma_m$ and $q \in Q$ we have always $H^A_{q,\sigma} \subseteq Q^m$.

For a tree t, we assume that all nodes are ordered in postorder and, thus, t is an *ordered tree.* Let $t[i]$ be the ith node of t and $\mathtt{des}(t[i])$ be the set of all descendants of $t[i]$ including $t[i]$ itself. Thus, $t[l(i) \ldots i]$ is the subtree rooted at $t[i]$, that is the subtree consisting of node i and all its descendants. Similarly, we define $\mathtt{anc}(t[i])$ to be the set of all ancestors of $t[i]$ including $t[i]$. The size $|t|$ of t is the number of nodes in t. We denote the character labeling a node $t[i]$ by $\sigma(i)$. Let θ be the empty tree. We say that $\mathtt{yield}(t)$ is a sequence of leaves in t. A forest is a sequence of trees and ordered when it has a left-to-right order among the trees. We only consider the ordered trees and the ordered forests in this paper. We refer the reader to the literature [4,6] for more details on tree automata.

3 Tree Edit-Distance

Given an alphabet Σ, let $\Omega = \{(a \to b) \mid a, b \in \Sigma \cup \{\lambda\}\} \setminus \{(\lambda, \lambda)\}$ be a set of edit operations. There are three edit operations: *deletions* $(a \to \lambda)$, *insertions* $(\lambda \to a)$ and *substitutions* $(a \to b)$ for $a \neq b$. We associate a non-negative edit cost to each edit operation $\omega_i \in \Omega$ as a function $\mathtt{C} : \Omega \to \mathbb{R}_+$. We assume that $\mathtt{C}$ is a distance metric satisfying the following conditions:

(i) $\mathtt{C}(a \to b) \geq 0, \mathtt{C}(a \to a) = 0$,
(ii) $\mathtt{C}(a \to b) = \mathtt{C}(b \to a)$ and
(iii) $\mathtt{C}(a \to c) \leq \mathtt{C}(a \to b) + \mathtt{C}(b \to c)$, where $a, b, c \in \Sigma \cup \{\lambda\}$.

An *edit script* $S \in \Omega^*$ between two trees t_1 and t_2 is a sequence of edit operations transforming t_1 into t_2. The cost $\mathtt{C}(S)$ of $S = s_1 s_2 \cdots s_n$ is $\mathtt{C}(S) = \sum_{i=1}^{n} \mathtt{C}(s_i)$. An *optimal edit script* between t_1 and t_2 is an edit script of minimum cost and the minimum cost is the *tree edit-distance* between t_1 and t_2.

Definition 1. *The tree edit-distance $d(t_1, t_2)$ of two trees t_1 and t_2 is*

$$d(t_1, t_2) = \min\{\mathtt{C}(S) \mid S \textit{ is an edit script transforming } t_1 \textit{ into } t_2\}.$$

That is, if S is an optimal edit script that transforms t_1 into t_2, then $\mathtt{C}(S) = d(t_1, t_2)$. We, in particular, consider the *top-down tree edit-distance*, which allows deletions and insertions of nodes only at leaves; namely, a node can be inserted or deleted only at leaf level. Thus, when an edit script S consists of edit operations where insertions or deletions occur only at leaf level, we say that S is a top-down edit script. We define the top-down tree edit-distance as follows:

Definition 2. *The top-down tree edit-distance $d(t_1, t_2)$ of two trees t_1 and t_2 is*

$$d(t_1, t_2) = \min\{\mathtt{C}(S) \mid S \textit{ is a top-down edit script transforming } t_1 \textit{ into } t_2\}.$$

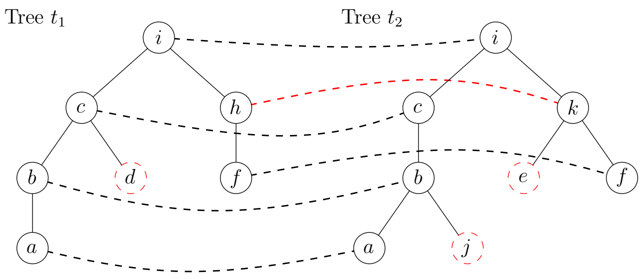

Fig. 1. A (top-down) mapping example between two trees t_1 and t_2

The example mapping in Fig. 1 depicts an edit script S that transforms t_1 into t_2 by deleting a node d, inserting two nodes j and e, and substituting a node h with k. Thus, the corresponding edit script S is

$$S = (a \to a)(\lambda \to j)(b \to b)(d \to \lambda)(c \to c)(\lambda \to e)(f \to f)(h \to k)(i \to i).$$

Let T_1 and T_2 be the sets of nodes in t_1 and t_2, respectively. We define a triple (M, t_1, t_2) to be a mapping from t_1 to t_2, where $M \subseteq T_1 \times T_2$ is a set of pair of nodes (i, j) for $1 \le i \le |t_1|$ and $1 \le j \le |t_2|$. We use M instead of (M, t_1, t_2) for simplicity when there is no confusion. We assume that trees are ordered in postorder. For any pair of (i_1, j_1) and (i_2, j_2) in M, the mapping M has the following restrictions:

(i) $i_1 = i_2$ if and only if $j_1 = j_2$ (one-to-one)
(ii) $i_1 < i_2$ if and only if $j_1 < j_2$ (sibling order preserved)
(iii) $t_1[i_1] \in \mathtt{anc}(t_1[i_2])$ if and only if $t_2[j_1] \in \mathtt{anc}(t_2[j_2])$ (ancestor order preserved)

We say that a node $t_1[i]$ is *touched by a line* if there exists a pair $(i, j) \in M$. Let I and J be the sets of nodes in t_1 and t_2, respectively, that are not touched by any line in M. Then, we define the cost $\mathtt{C}(M)$ of M to be

$$\mathtt{C}(M) = \sum_{(i,j)\in M} \mathtt{C}(\sigma(i) \to \sigma(j)) + \sum_{i \in I} \mathtt{C}(\sigma(i) \to \lambda) + \sum_{j \in J} \mathtt{C}(\lambda \to \sigma(j)).$$

Next we extend the concept of the edit-distance as a distance metric between a tree and a tree language.

Definition 3. *We define the edit-distance $d(t, L)$ between a tree t and a tree language L over Σ to be*

$$d(t, L) = \inf\{d(t, t') \mid t' \in L\}.$$

Then, we define the edit-distance between two tree languages as follows:

Definition 4. *We define the edit-distance $d(L, R)$ between two tree languages L and R over Σ to be*

$$d(L, R) = \inf\{d(t, t') \mid t \in L \text{ and } t' \in R\}.$$

In other words, the edit-distance between L and R is the minimum edit-distance between the most similar pair of trees from two tree languages. Note that these distance measures are symmetric. Thus, $d(t, t') = d(t', t)$, $d(t, L) = d(L, t)$, and $d(L, R) = d(R, L)$.

4 Edit-Distance of Regular Tree Languages

Before we tackle the main problem, we introduce *k-bounded tree automata*, which have more expressive power than ranked TAs. Note that if we insert or delete a node in a ranked tree, then it does not preserve its rank anymore. Therefore, instead of considering ranked TAs, we use TAs that allow only a finite number of children.

Definition 5. *A* k-bounded tree automaton *is specified by a tuple*

$$A = (\Sigma, Q, F, \delta),$$

where Σ is an alphabet, Q is a finite set of states, $F \subseteq Q$ is a set of final states, and δ associates to each $\sigma \in \Sigma$ a mapping $\sigma_g : Q^{\le k} \to 2^Q$.

We define the size $|A|$ of a k-bounded TA A to be $|Q| + \sum_{\sigma_\delta(q_1,\ldots,q_l)\to q}(l+1)$. A k-bounded TA is, thus, an unranked TA where there exists a constant k such that any node can have at most k children. Note that a ranked TA is a restricted variant of a k-bounded TA. If Σ is a ranked alphabet and $k = \max\{r(\sigma) \mid \sigma \in \Sigma\}$, then any ranked TA over Σ is k-bounded.

We introduce a polynomial algorithm for computing the tree edit-distance between two regular tree languages L and R described by k-bounded TAs. By Definition 4, the edit-distance between L and R is the edit-distance between two closest pair of trees $t \in L$ and $t' \in R$.

Let $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$ be two k-bounded TAs accepting regular tree languages L and R, respectively. From A, let A_q be a new TA that has a unique final state $q \in Q$. For simplicity, we denote $d(L(A_q), L(B_{q'}))$ by $d(q, q')$. Then, we have the following statement.

Proposition 6. *Let $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$ be two k-bounded TAs accepting two tree languages L and R. Then,*

$$d(L, R) = \min\{d(q, q') \mid q \in F, q' \in F'\}.$$

Proposition 6 states that we can compute the edit-distance between two regular tree languages by computing the minimum edit-distance between a pair (q, q') of states q, q', where q is a final state of A and q' is a final state of B. We show that it is possible to compute such distances in polynomial time by recursively computing the distance between all pairs of states from A and B. Before we give the main algorithm, we first define the edit-distance between two sequences of states. We denote the minimum edit-distance between two forests from two sequences of states $q_1q_2\ldots q_i$ and $q'_1q'_2\ldots q'_j$ by $d(q_1q_2\ldots q_i, q'_1q'_2\ldots q'_j)$. From now, we denote the sequence $q_1q_2\ldots q_i$ of states by $\mathbb{S}_{1,i}$ and the sequence $q'_1q'_2\ldots q'_j$ by $\mathbb{S}'_{1,j}$. In other words, $\mathbb{S}_{m,n} \in Q^{\leq k}$ and $\mathbb{S}'_{m,n} \in (Q')^{\leq k}$ for $m \leq n$.

Given two k-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, we define the following sets for the edit-distance $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ between two sequences of states.

(i) $I(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j-1}) + d(\lambda, \mathbb{T}') + \mathtt{C}(\lambda, \sigma') \mid \sigma'_{\delta'}(\mathbb{T}') = q'_j\}$,
(ii) $D(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j}) + d(\mathbb{T}, \lambda) + \mathtt{C}(\sigma, \lambda) \mid \sigma_\delta(\mathbb{T}) = q_i\}$, and
(iii) $S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j-1}) + d(\mathbb{T}, \mathbb{T}') + \mathtt{C}(\sigma, \sigma') \mid \sigma_\delta(\mathbb{T}) = q_i, \sigma'_{\delta'}(\mathbb{T}') = q'_j\}$,

where $\mathbb{T} \in Q^*$ and $\mathbb{T}' \in (Q')^*$. Let I, D, and S denote insertion, deletion and substitution, respectively. Based on the three sets, we present a recursive equation for computing the edit-distance between two forests accepted by two sequences of states from A and B as follows:

Lemma 7. *Given two k-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, the edit-distance $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ is defined as*

$$\min[I(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) \cup D(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) \cup S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})],$$

where $\mathbb{S}_{1,i} \in Q^{\leq k}$ and $\mathbb{S}'_{1,j} \in (Q')^{\leq k}$.

We observe that the computation of $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ may have a self-dependency problem in the computation. For example, consider the set $S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$, which is the 3rd case in the proof of Lemma 7.

$$S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j-1}) + d(\mathbb{T}, \mathbb{T}') + \mathtt{C}(\sigma, \sigma') \mid \sigma_\delta(\mathbb{T}) = q_i, \sigma'_{\delta'}(\mathbb{T}') = q'_j\}.$$

Then, the computation of $S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ requires the value of $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ when $\mathbb{T} = \mathbb{S}_{1,i}$ and $\mathbb{T}' = \mathbb{S}'_{1,j}$. This implies that we need the value of $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ for computing the value of $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$. We solve this dependency problem by using induction on the height of the optimal mapping. Assume that we have two trees t_1, t_2 and an optimal mapping $M \subseteq t_1 \times t_2$. We construct a new mapping M' from M by removing all insertions and deletions. Then, the resulting mapping M' consists of the pairs $(i, j) \in t_1 \times t_2$ such that $i \neq \lambda$ and $j \neq \lambda$. We call M' the *trimmed mapping*. Now we define the *height n edit-distance* to be the edit-distance between two trees, where the height of the corresponding optimal trimmed mapping for the edit-distance is at most n. Let $d_n(q, q')$ be the *height n edit-distance* between two states q and q'. Note that the similar notation can be used for the height n edit-distance between two sequences of states such as $d_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$.

Then we define the following sets for recurrence of the height n edit-distance between two sequences of states.

(i) $I_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j-1}) + d_0(\lambda, \mathbb{T}') + \mathtt{C}(\lambda, \sigma') \mid \sigma'_{\delta'}(\mathbb{T}') = q'_j\}$,
(ii) $D_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d_n(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j}) + d_0(\mathbb{T}, \lambda) + \mathtt{C}(\sigma, \lambda) \mid \sigma_\delta(\mathbb{T}) = q_i\}$, and
(iii) $S_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d_n(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j-1}) + d_{n-1}(\mathbb{T}, \mathbb{T}') + \mathtt{C}(\sigma, \sigma') \mid \sigma_\delta(\mathbb{T}) = q_i, \sigma'_{\delta'}(\mathbb{T}') = q'_j\}$.

Then, $d_n(q, q')$ should be the minimum of the following sets:

(i) $I_n(q, q') = \{d_0(q, \lambda) + d_0(\lambda, \mathbb{T}') + \mathtt{C}(\lambda, \sigma') \mid \sigma'_{\delta'}(\mathbb{T}') = q'\}$,
(ii) $D_n(q, q') = \{d_0(\lambda, q') + d_0(\mathbb{T}, \lambda) + \mathtt{C}(\sigma, \lambda) \mid \sigma_\delta(\mathbb{T}) = q\}$, and
(iii) $S_n(q, q') = \{d_{n-1}(\mathbb{T}, \mathbb{T}') + \mathtt{C}(\sigma, \sigma') \mid \sigma_\delta(\mathbb{T}) = q, \sigma'_{\delta'}(\mathbb{T}') = q'\}$.

Note that the following equations hold:

– $d_n(q, q') = \infty$ if $n < 0$,
– $d_0(\lambda, \lambda) = 0$,
– $d_0(q, \lambda) = \min\{|t| \mid t \in L(A_q)\}$, and
– $d_0(q, q') = \min\{|t| \mid t \in L(A_q)\} + \min\{|t'| \mid t' \in L(B_{q'})\}$.

Now we establish the following lemma.

Lemma 8. *Given two k-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, the height n edit-distance $d_n(q, q')$ is defined as*

$$\min[d_{n-1}(q, q') \cup I_n(q, q') \cup D_n(q, q') \cup S_n(q, q')],$$

where $q \in Q$, $q' \in Q'$ and $n \geq 0$.

Proof. We prove by induction on n. First, we start with the case when $n = 0$. Consider a mapping M between two trees $t \in L(A_q)$ and $t' \in L(B_{q'})$, where $|t| = i$ and $|t'| = j$. We construct the mapping M in postorder, therefore, the mapping between two root nodes of t and t' should be the last to consider. By the definition, the height of the trimmed mapping of M should be 0. This implies that there is no mapping for substitutions between two trees. Since $d_{-1}(q, q')$ is ∞, we compute the case when $n = 0$ by only considering insertions and deletions.

Assume that the case $n = l$ holds. Then, we prove that the case $n = l + 1$ also holds. By the assumption, we know that the case when the height of the trimmed mapping is lower than $l+1$ is considered by the first term $d_l(q, q')$. Now we should prove for the case when the height of the trimmed mapping is $l + 1$. There are three cases to consider:

(i) $t[i]$ is not touched by a line in M. Then, we have $(i, \lambda) \in M$. Note that insertions and deletions do not change the height of the trimmed mapping. Therefore,

$$D_n(q, q') = \min\{d_0(\lambda, q') + d_0(\mathbb{T}, \lambda) + \mathtt{C}(\sigma, \lambda) \mid \sigma(\mathbb{T}) = q \in \delta\}.$$

(ii) $t'[j]$ is not touched by a line in M. Then, we have $(\lambda, j) \in M$. Symmetrically,

$$I_n(q, q') = \min\{d_0(q, \lambda) + d_0(\lambda, \mathbb{T}') + \mathtt{C}(\lambda, \sigma') \mid \sigma'(\mathbb{T}') = q' \in \delta'\}.$$

(iii) $t[i]$ and $t'[j]$ are touched by lines in M. Thus, $(i, j) \in M$ by the mapping restrictions. That means we need an optimal mapping between two forests $t[1 \ldots i - 1]$ and $t'[1 \ldots j - 1]$. The height of optimal trimmed mapping between these two forests is l. Therefore,

$$S_n(q, q') = \min\{d_{n-1}(\mathbb{T}, \mathbb{T}') + \mathtt{C}(\sigma, \sigma') \mid \sigma(\mathbb{T}) = q \in \delta, \sigma'(\mathbb{T}') = q' \in \delta'\}.$$

Since all possible optimal mappings between two trees can be computed by the definition, we prove the lemma. □

One remaining issue is how many times we should iterate the computation of recurrence for computing the correct edit-distance between two regular tree languages. We can show that $|Q||Q'|$ iterations are enough for computing the edit-distance between two regular languages as the height of the optimal trimmed mapping is at most $|Q||Q'|$ to avoid the repetition of the same state pair. We establish the following result.

Lemma 9. *Given two k-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, $d_{mn}(q, q') = d(q, q')$, where $q \in Q$, $q' \in Q'$, $m = |Q|$ and $n = |Q'|$.*

We analyze the time complexity of Algorithm 1 for computing the top-down tree edit-distance between two regular languages given by two k-bounded TAs and establish the following result.

Theorem 10. *Let $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$ be two k-bounded TAs. Then, we can compute the edit-distance $d(L(A), L(B))$ in $\mathcal{O}(m^2 n^2)$ time, where $m = |A|$ and $n = |B|$.*

Algorithm 1. Computing $d(L(A), L(B))$

```
input  : Two $k$-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma, Q', F', \delta')$
output : $d(L(A), L(B))$

$d_0(\lambda, \lambda) \leftarrow 0$;
for $q \in Q$ do
    $d_0(q, \lambda) \leftarrow \min\{|t| \mid t \in L(A_q)\}$;
end
for $q' \in Q'$ do
    $d_0(\lambda, q') \leftarrow \min\{|t'| \mid t' \in L(B_{q'})\}$;
end
for $i \leftarrow 0$ to $|Q||Q'|$ do
    for $q \in Q$ do
        for $q' \in Q'$ do
            $d_i(q, q') \leftarrow \min[d_{i-1}(q, q') \cup I_i(q, q') \cup D_i(q, q') \cup S_i(q, q')]$;
        end
    end
end
return $\min\{d_{|Q||Q'|}(q, q') \mid q \in F, q' \in F'\}$;
```

5 Unbounded Case

It is well known that unranked tree automata describe regular tree languages of unranked and unbounded trees, which are the generalizations of regular tree languages of ranked and bounded trees [4]. We generalize the edit-distance computation between two regular tree languages accepted by k-bounded TAs to the unbounded case in this section.

Unlike in a k-bounded or ranked TA, we have a regular language over the state set Q called a horizontal language instead of a sequence of states in an unranked TA. Therefore, we compute the minimum edit-distance between two forests accepted by two sequences of states from two horizontal languages. Let $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$ be two unranked TAs. Then, the edit-distance between two states $q \in Q$ and $q' \in Q'$ is defined as the minimum of the following three sets.

(i) $I(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j-1}) + d(\lambda, \mathbb{T}') + \mathtt{C}(\lambda, \sigma') \mid \sigma'_{\delta'}(\mathbb{T}') = q'_j, \mathbb{T}' \in L(H^B_{q',\sigma'})\}$,

(ii) $D(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j}) + d(\mathbb{T}, \lambda) + \mathtt{C}(\sigma, \lambda) \mid \sigma_\delta(\mathbb{T}) = q_k, \mathbb{T} \in L(H^A_{q,\sigma})\}$, and

(iii) $S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j-1}) + d(\mathbb{T}, \mathbb{T}') + \mathtt{C}(\sigma, \sigma') \mid \sigma_\delta(\mathbb{T}) = q, \sigma'_{\delta'}(\mathbb{T}') = q', \mathbb{T} \in L(H^A_{q,\sigma}), \mathbb{T}' \in L(H^B_{q',\sigma'})\}$.

Similarly to the bounded case, we define the recurrence for the edit-distance between two forests accepted by two sequences of states from A and B as follows:

Lemma 11. *Given two unranked TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, the height n edit-distance $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ is defined as*

$$\min[I(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) \cup D(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) \cup S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})],$$

where $\mathbb{S}_{1,i} \in Q^*$ and $\mathbb{S}'_{1,j} \in (Q')^*$.

Now we consider the runtime for computing the edit-distance between two unranked TAs. The main difference compared with the bounded case is that we need to compute the edit-distance between two forests accepted by two regular horizontal languages instead of two fixed sequences of states.

Corollary 12 (Mohri [12]). *Given two FAs A and B, we can compute the edit-distance $d(L(A), L(B))$ in $\mathcal{O}(mn \cdot \log mn)$ time, where $m = |A|$ and $n = |B|$.*

Theorem 13. *Let $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$ be two unranked TAs. Then, we can compute the edit-distance $d(L(A), L(B))$ in $\mathcal{O}(m^2 n^2 \cdot \log mn)$ time, where $m = |A|$ and $n = |B|$.*

6 An Application of the Tree Edit-Distance Problem

It is known that the edit-distance between two context-free languages is not computable [12]. Moreover, the emptiness of the intersection of two context-free languages is also undecidable. Now we show that it is possible to check whether or not two CFGs have a common string whose derivation trees are structurally equivalent by relying on the edit-distance computation between two regular tree languages.

Proposition 14 (Comon et al. [4]). *The following statements hold.*

- *Given a context-free grammar G, the set of derivation trees of $L(G)$ is a regular tree language.*
- *Given a regular tree language L,* `yield`(L) *is a context-free language.*
- *There exists a regular tree language that is not a set of derivation trees of a context-free language.*

Based on Proposition 14, we establish the following result.

Lemma 15. *Given two CFGs G and G', we can determine whether or not there exists a common string $w \in L(G) \cap L(G')$ whose derivation trees from G and G' are structurally equivalent.*

7 Conclusions

We have studied the top-down tree edit-distance between two regular tree languages. The tree edit-distance between two tree languages is the minimum tree edit-distance between two trees from two languages. We, in particular, have considered the restricted version of the general tree edit-distance problem called the top-down tree edit-distance. We have proposed an $\mathcal{O}(m^2n^2)$ algorithm for computing the edit-distance between two regular tree languages given by two k-bounded TAs of sizes m and n. For the edit-distance between two unranked TAs of sizes m and n, we have designed an $\mathcal{O}(m^2n^2 \log mn)$ algorithm.

Given two CFGs G and G', we have also shown that it is decidable to determine whether or not there exists a common string whose derivation trees from G and G' are structurally equivalent using the proposed algorithm.

References

1. Bunke, H.: Edit distance of regular languages. In: Proceedings of the 5th Annual Symposium on Document Analysis and Information Retrieval, pp. 113–124 (1996)
2. Chawathe, S.S.: Comparing hierarchical data in external memory. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 90–101 (1999)
3. Choffrut, C., Pighizzini, G.: Distances between languages and reflexivity of relations. Theoretical Compututer Science 286(1), 117–138 (2002)
4. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (2007), `http://www.grappa.univ-lille3.fr/tata` (release October 12, 2007)
5. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. ACM Transactions on Algorithms 6(1), 2:1–2:19 (2009)
6. Gécseg, F., Steinby, M.: Tree languages. In: Handbook of Formal Languages, Vol. 3: Beyond Words, pp. 1–68. Springer-Verlag New York, Inc. (1997)
7. Hamming, R.W.: Error Detecting and Error Correcting Codes. Bell System Technical Journal 26(2), 147–160 (1950)
8. Han, Y.-S., Ko, S.-K., Salomaa, K.: Computing the edit-distance between a regular language and a context-free language. In: Yen, H.-C., Ibarra, O.H. (eds.) DLT 2012. LNCS, vol. 7410, pp. 85–96. Springer, Heidelberg (2012)
9. Han, Y.-S., Ko, S.-K., Salomaa, K.: Approximate matching between a context-free grammar and a finite-state automaton. In: Konstantinidis, S. (ed.) CIAA 2013. LNCS, vol. 7982, pp. 146–157. Springer, Heidelberg (2013)
10. Klein, P.N.: Computing the edit-distance between unrooted ordered trees. In: Proceedings of the 6th Annual European Symposium on Algorithms, pp. 91–102 (1998)
11. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
12. Mohri, M.: Edit-distance of weighted automata: General definitions and algorithms. International Journal of Foundations of Computer Science 14(6), 957–982 (2003)
13. Myers, G.: Approximately matching context-free languages. Information Processing Letters 54, 85–92 (1995)
14. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in XML documents. In: Proceedings of the 5th International Workshop on the Web and Databases, pp. 61–66 (2002)

15. Reis, D.C., Golgher, P.B., Silva, A.S., Laender, A.F.: Automatic web news extraction using tree edit distance. In: Proceedings of the 13th International Conference on World Wide Web, pp. 502–511 (2004)
16. Selkow, S.: The tree-to-tree editing problem. Information Processing Letters 6(6), 184–186 (1977)
17. Tai, K.C.: The tree-to-tree correction problem. Journal of the ACM 26(3), 422–433 (1979)
18. Tekli, J., Chbeir, R., Yetongnon, K.: Survey: An overview on XML similarity: Background, current trends and future directions. Computer Science Review 3(3), 151–173 (2009)
19. Wagner, R.A.: Order-n correction for regular languages. Communications of the ACM 17, 265–268 (1974)
20. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. Journal of the ACM 21, 168–173 (1974)
21. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research, pp. 354–359 (1990)
22. Yang, R., Kalnis, P., Tung, A.K.H.: Similarity evaluation on tree-structured data. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 754–765 (2005)
23. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing 18(6), 1245–1262 (1989)
24. Zhang, Z., Cao, R.L.S., Zhu, Y.: Similarity metric for XML documents. In: Proceedings of Workshop on Knowledge and Experience Management (2003)