Top-Down Tree Edit-Distance of Regualr Tree Languages

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

[1]Department of Computer Science
Yonsei University

[2]School of Computing
Queen's University

LATA 2014, Madrid

Overview

1. Introduction
2. Problem : compute top-down tree edit-distance of RTLs
3. Algorithm : dynamic programming approach
4. Conclusions

Basic Definitions

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:

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

Basic Definitions

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

$$\mathtt{C}(S) = \sum_{i=1}^{n} \mathtt{C}(s_i).$$

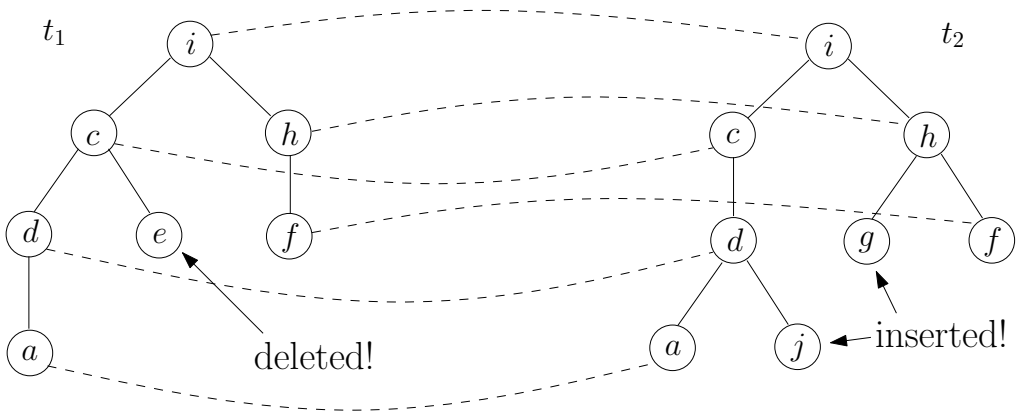
An **optimal edit script** between t_1 and t_2 is an edit script of minimum cost and the minimum cost is the **tree edit-distance** between t_1 and t_2.

Tree Edit-Distance

Definition

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

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



That is, if S is an optimal edit script that transforms t_1 into t_2, then

$$\mathtt{C}(S) = d(t_1, t_2).$$

Tree Edit-Distance Algorithms

Previous results

	Time	Space
Tai [1979]	$O(n^3m^3)$	$O(n^3m^3)$
Shasha and Zhang [1989]	$O(n^2m^2)$	$O(nm)$
Klein [1998]	$O(m^2n \log n)$	$O(nm)$
Chen [2001]	$O(nm^{2.5})$	$O(n^3)$
Demaine et al' [2009]	$O(nm^2(1 + \log n/m))$	$O(nm)$

It is proven that $O(n^3)$ is the worst-case optimal.

Top-Down Tree Edit-Distance

We, in particular, consider the

top-down tree edit-distance,

which allows deletions and insertions of nodes only **at leaves**!

Definition

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

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

Selkow [1977] designed an $O(mn)$ time algorithm.

Applications of Top-Down Tree Edit-Distance

1. Automatic web news extraction,
2. Comparison of hierarchical data in external memory,
3. XML document clustering,
4. and so on...

Edit-Distance of Regular Tree Languages

Definition

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

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 **minimum edit-distance between the most similar pair of trees** from two tree languages.

k-Bounded Tree Automaton

Definition

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 $\sigma_g : Q^{\leq k} \to 2^Q$.

A k-bounded TA is an **unranked TA** where there exists a constant k such that any node has at most k children.

ranked TA $\subseteq$ 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.

Main Idea

Given a TA $A = (\Sigma, Q, F, \delta)$ and a state $q \in Q$, A_q is a TA with a single final state q. For simplicity, denote $d(L(A_q), L(B_{q'}))$ by $d(q, q')$.

Lemma

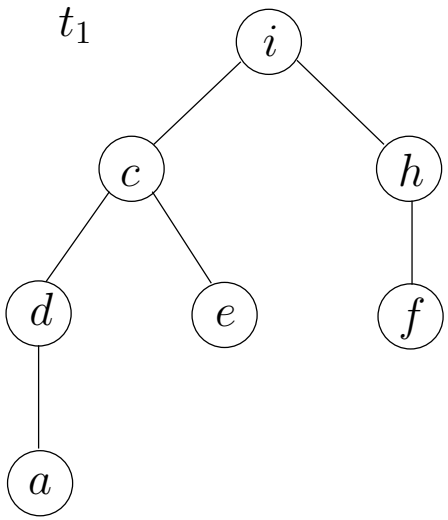
Given two k-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$ accepting two tree languages L and R, respectively,

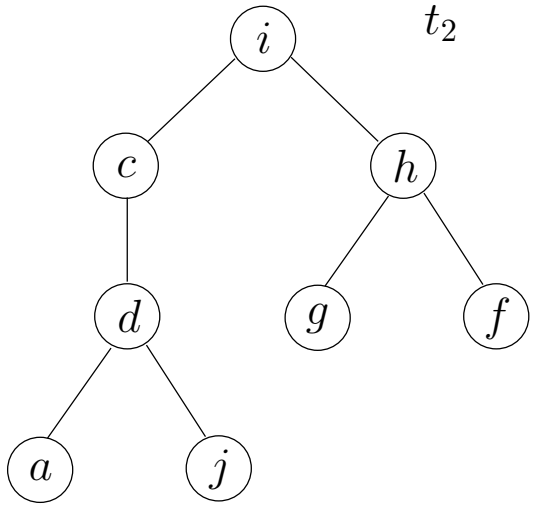
$$d(L, R) = \min\{d(q, q') \mid q \in F, q' \in F'\}.$$

Question

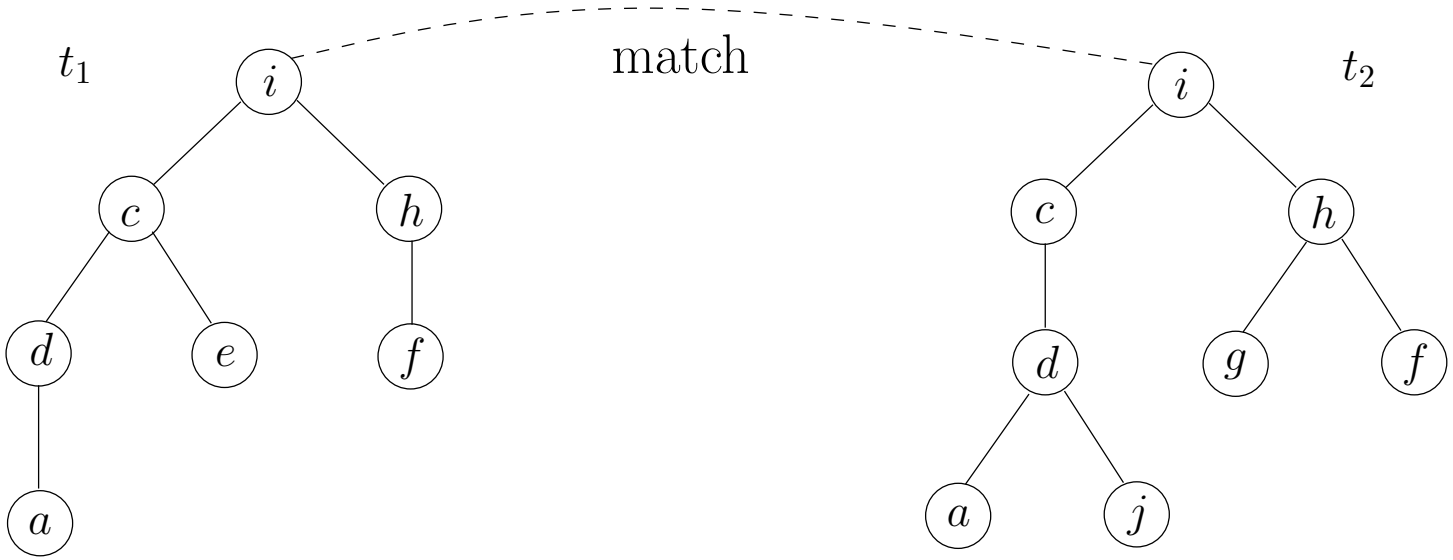
How to compute $d(q, q')$ for all state pairs q and q'?

Main Idea



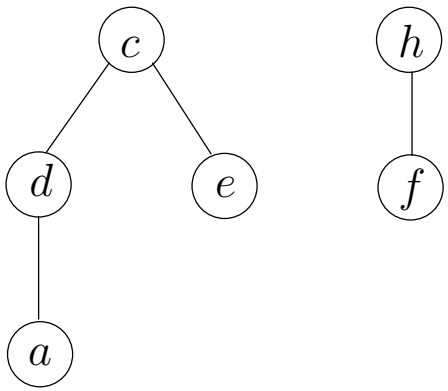


Main Idea

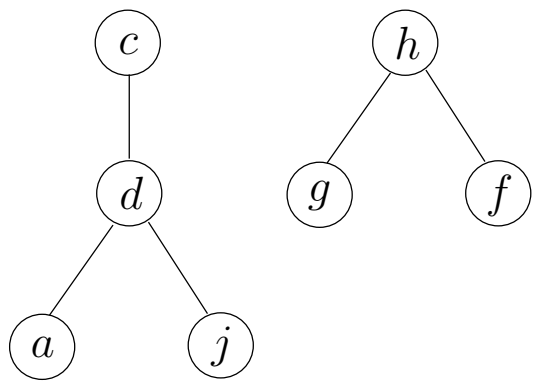


Main Idea

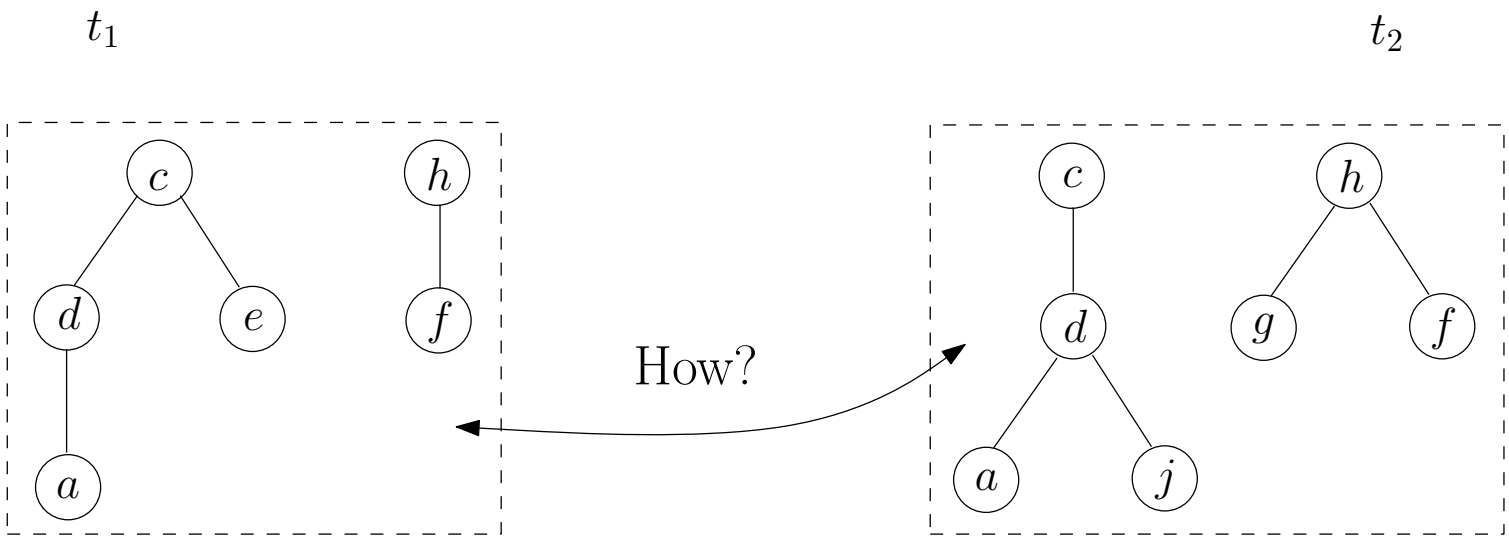
t_1



t_2



Main Idea



Distance between Forests (Hedges)

Notation - sequence of states

Denote the sequence $q_1 q_2 \dots q_i$ of states by $\mathbb{S}_{1,i}$ and the sequence $q'_1 q'_2 \dots q'_j$ by $\mathbb{S}'_{1,j}$.

The edit-distance $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ between two sequences of states.

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

where $\mathbb{T} \in Q^*$ and $\mathbb{T}' \in (Q')^*$.

Main Lemma

Lemma

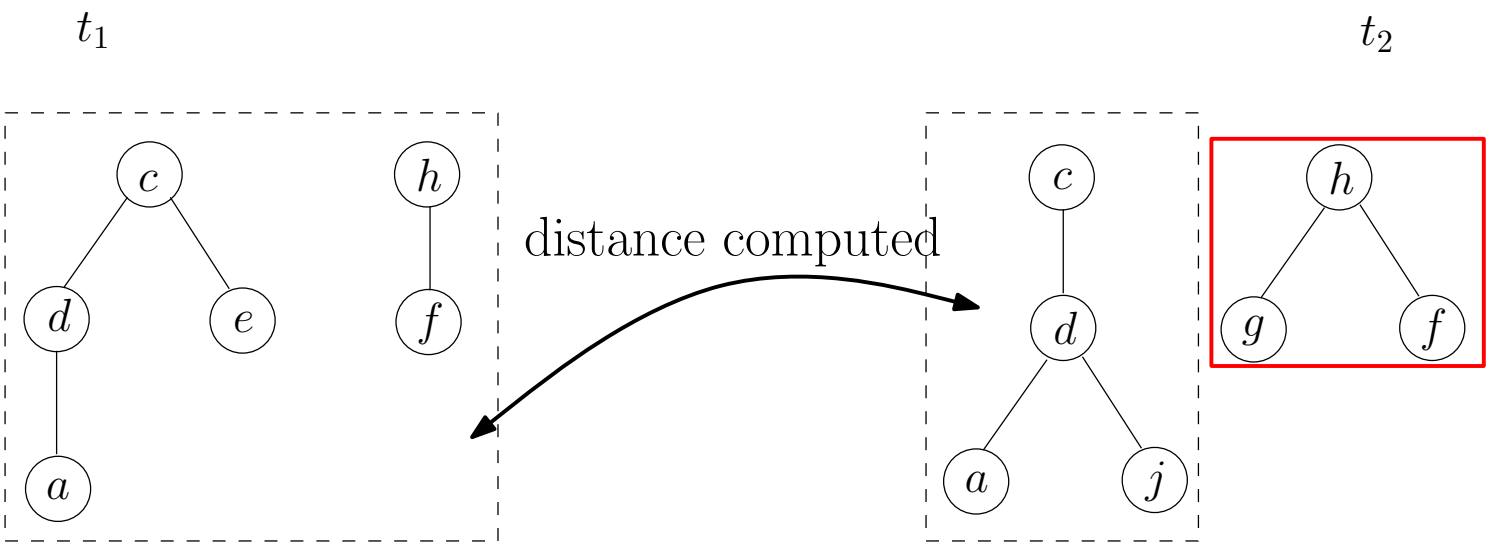
Given two k-bounded TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, the edit-distance $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ is defined as

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

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

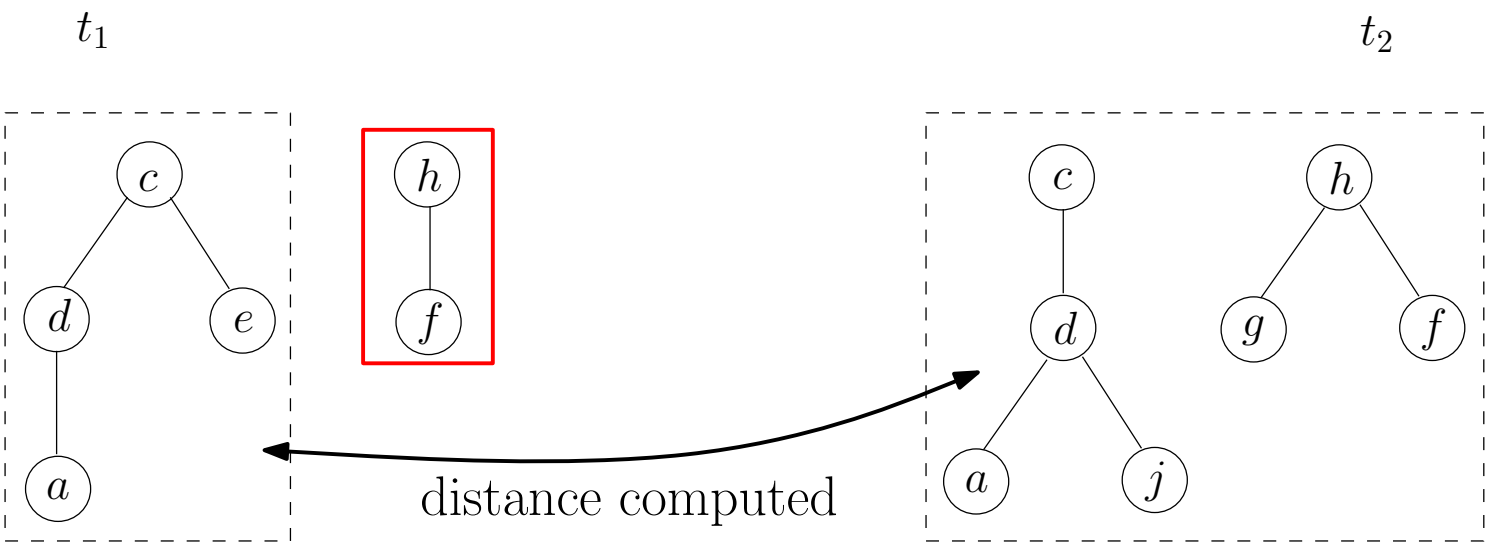
Case (i) - Insertion

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



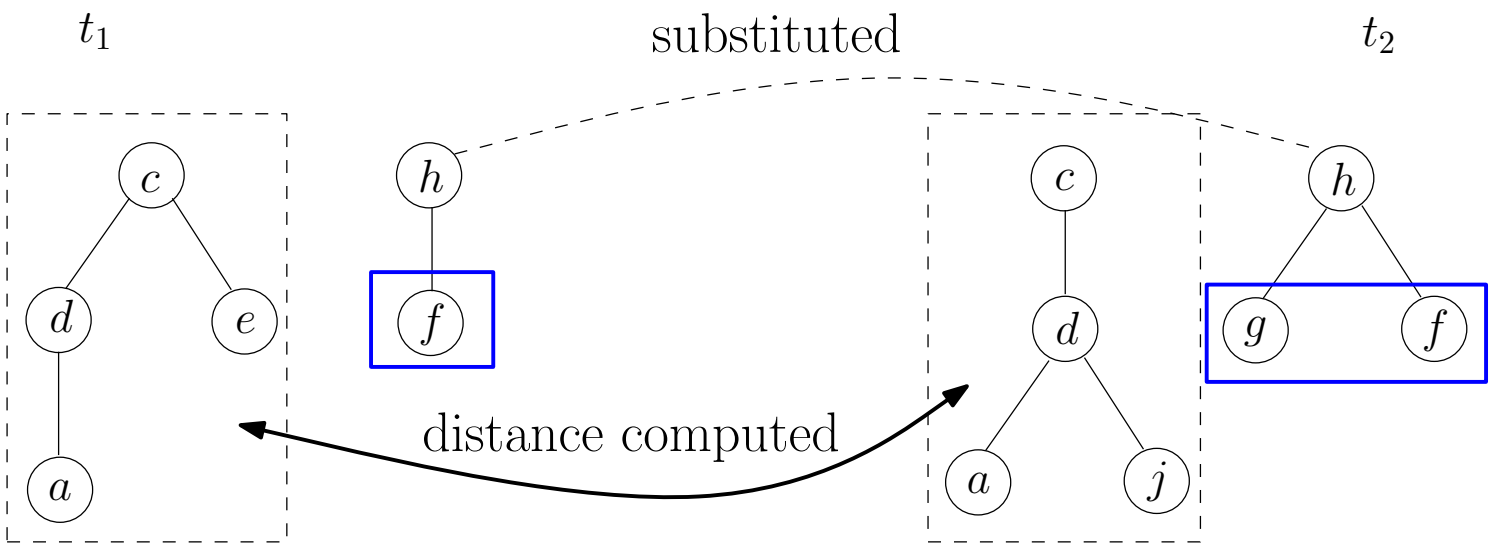
Case (ii) - Deletion

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



Case (iii) - Substitution

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



Dependency Problem

Consider the computation of $S(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$, which is

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

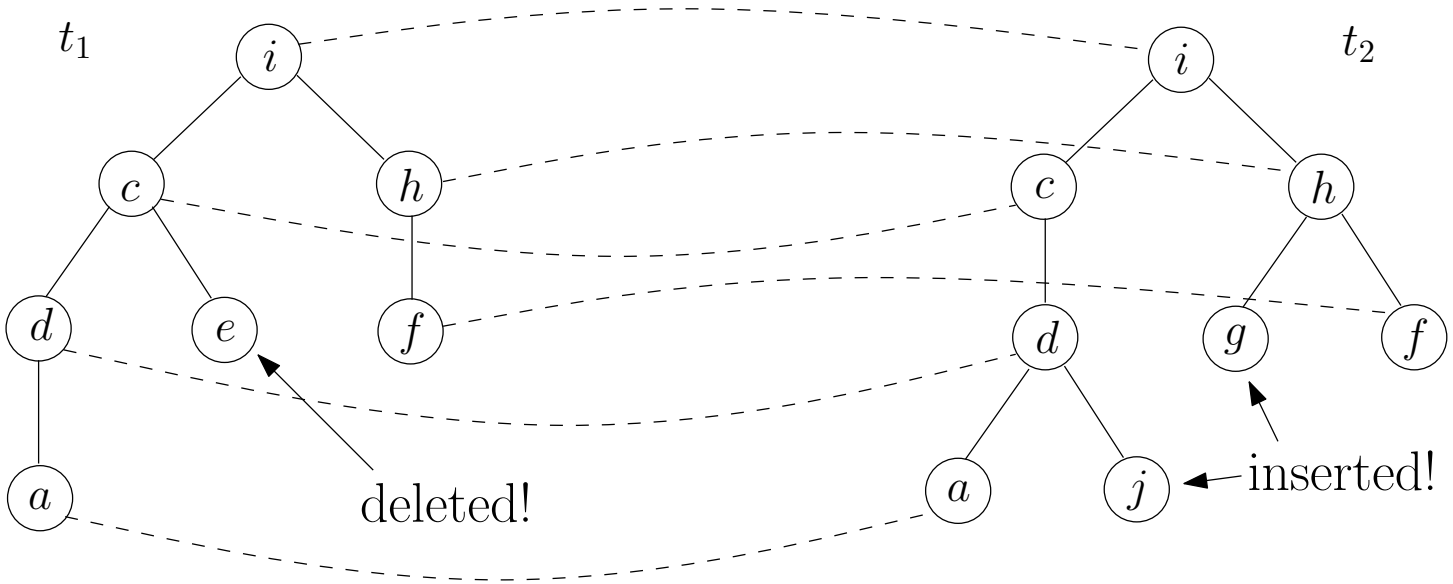
If $\mathbb{T} = \mathbb{S}_{1,i}$ and $\mathbb{T}' = \mathbb{S}'_{1,j}$? We need the value of $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$ for computing the value of $d(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j})$.

Dependency problem!

Trimmed Mapping

Assume that we have two trees t_1, t_2 and an optimal mapping $M \subseteq t_1 \times t_2$.

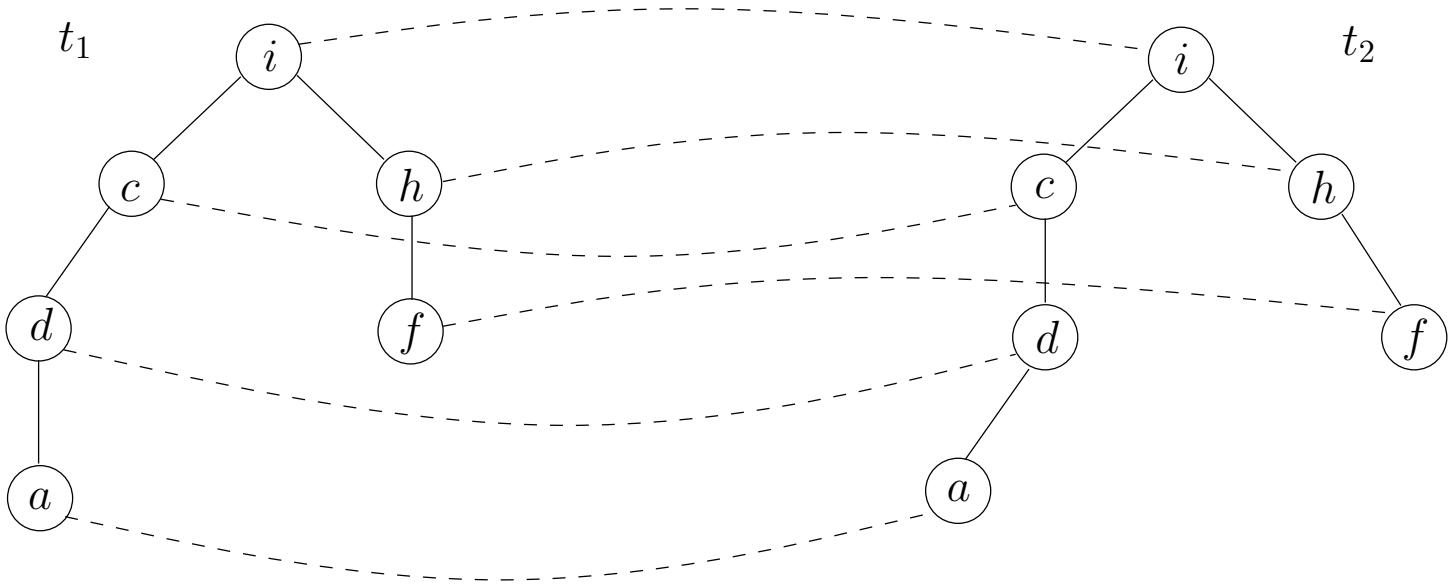
We construct a new mapping M' from M by removing all insertions and deletions.



Trimmed 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.



Height n Edit-Distance

Definition ($d_n(q, q')$)

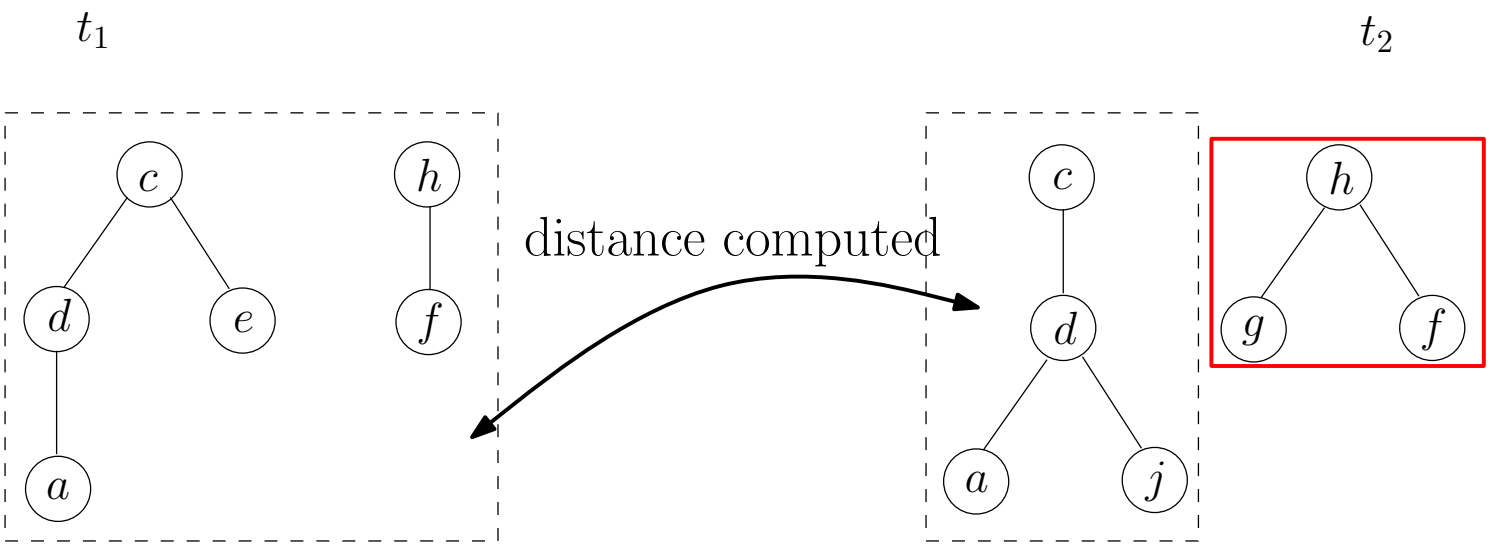
The edit-distance between two trees where the height of the optimal trimmed mapping for the edit-distance is at most n.

Then we define the following sets:

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

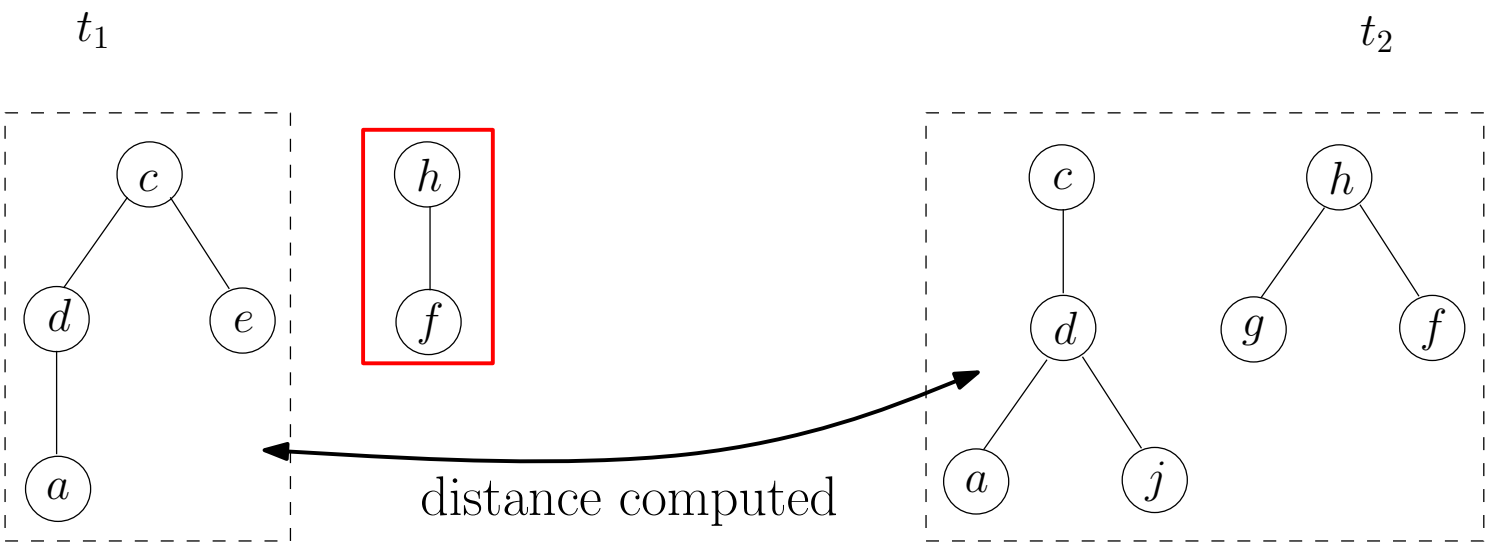
Case (i) - Insertion

$$I_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j-1}) + d_0(\lambda, \mathbb{T}') + \mathtt{C}(\lambda, \sigma') \mid \sigma'_{\delta'}(\mathbb{T}') = q'_j\}$$



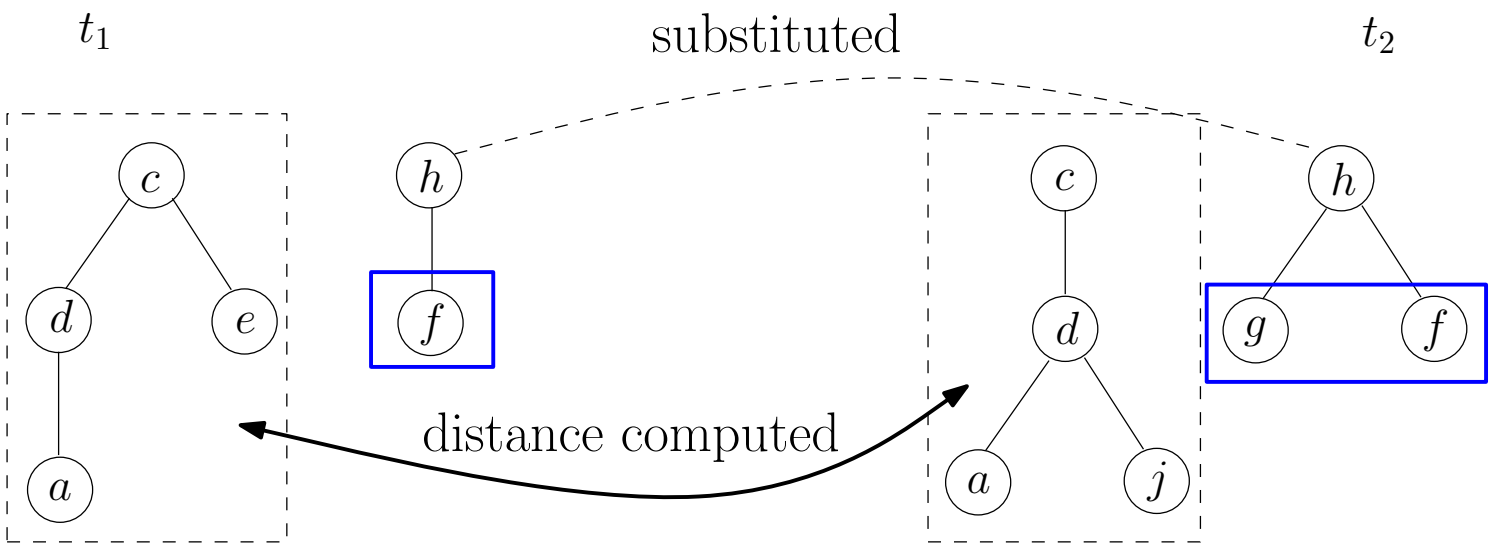
Case (ii) - Deletion

$$D_n(\mathbb{S}_{1,i}, \mathbb{S}'_{1,j}) = \{d_n(\mathbb{S}_{1,i-1}, \mathbb{S}'_{1,j}) + d_0(\mathbb{T}, \lambda) + \mathtt{C}(\sigma, \lambda) \mid \sigma_\delta(\mathbb{T}) = q_i\}$$



Case (iii) - Substitution

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



Algorithm

1: $d_0(\lambda, \lambda) \leftarrow 0$;
2: **for** $q \in Q$ and $i = 0$ **to** $|Q|$ **do**
3: $\quad d_i(q, \lambda) \leftarrow \min\{|t| \mid t \in L(A_q), |t| \leq i\}$;
4: **end for**
5: **for** $q' \in Q'$ and $i = 0$ **to** $|Q'|$ **do**
6: $\quad d_i(\lambda, q') \leftarrow \min\{|t'| \mid t' \in L(B_{q'}), |t'| \leq i\}$;
7: **end for**
8: **for** $i = 0$ **to** $|Q||Q'|$ **do**
9: $\quad$ **for** $q \in Q$ and $q' \in Q'$ **do**
10: $\quad\quad d_i(q, q') \leftarrow \min[d_{i-1}(q, q') \cup I_i(q, q') \cup D_i(q, q') \cup S_i(q, q')]$;
11: $\quad$ **end for**
12: **end for**
13: **return** $\min\{d_{|Q||Q'|}(q, q') \mid q \in F, q' \in F'\}$;

Our Result

Now, the question is **how many times** we should iterate the computation.

The answer is mn.

Lemma

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 $n = |Q'|$.

As a result,

Theorem

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

Unranked Case

In unranked case, we have two horizontal languages instead of two sequences of states.

Corollary (Mohri, 2003)

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 $n = |B|$.

Therefore,

Theorem

Given two unranked TAs $A = (\Sigma, Q, F, \delta)$ and $B = (\Sigma', Q', F', \delta')$, 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 $n = |B|$.

Structural Equivalence of CFGs

Known result (Comon et al., 2007)

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, $\texttt{yield}(L)$ is a context-free language.

Lemma

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.

Conclusions

Results

- We have considered the problem of computing top-down tree edit distance between two regular tree languages.
- We have presented algorithms for computing top-down tree edit-distance
  - between two ranked k-bounded tree automata, and
  - between two unranked tree automata.

Future works

It will be interesting to see if we can

- compute the general tree edit-distance between two RTLs.
- design more efficient algorithms for the same problem.
- compute the similarity between XML schema instances.


감사합니다
Natick
Grazie
Danke
Ευχαριστίες
Dalu
Obrigado
Thank You
Köszönöm
Tack
Спасибо
Dank
Gracias
谢谢
Merci
Seé
ありがとう