社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－MPS－54（3）
2005／5／10

# 構造付き文字列のアラインメントに対する文法論的アプローチ

関　新之助　　小林　聡
電気通信大学大学院 電気通信学研究科 情報工学専攻

本論文は、木文法 (TAG) の構造的曖昧性の問題を解決する理論的枠組みを提供する。これは特に RNA2 次構造に代表される構造付き文字列から共通構造をアラインメント手法で抽出し、結果を TAG でモデリングする際に問題となる。

# A Grammatical Approach to the Alignment of SA-Strings

Shinnosuke Seki,　　Satoshi Kobayashi,
Department of Computer Science,
University of Electro-Communications,

In this paper, we are concerned with a structural ambiguity problem of tree adjoining grammars(TAGs), which is an important and essential problem when we try to model consensus structures of given set of ribonucleic acid(RNA) secondary structures by TAGs.

## 1 Introduction

*Tree adjoining grammars* (TAGs) were originally proposed as *tree-generating* systems by linguistic considerations. Among several features of TAGs, a descriptive capability of TAGs is of most interest to us. The idea that the symbols in a string derived at the same time in a derivation process forms a structure on the string has been utilized to model strings with structural information by grammars. We call such strings *structure-annotated strings*(*SA-strings*). The reason why TAGs have especially attracted great attention in broad fields is the ability of TAGs to describe certain discontinuous structures. For example, a string such as $xuyvx^Rwy^R$, where $u$, $v$, $w$, $x$ and $y$ are strings and $x^R(y^R)$ denotes the reversal of $x(y)$, can be readily modeled by a derived tree of TAGs, but not expressed by any context-free grammars(CFGs). The structural feature of this example is called *crossing dependency*, and this extra power of TAGs seems to be just the right kind for a certain practical application of modeling bioinformatical sequence data.

Ribonucleic acids(RNAs) form secondary structures (2nd-structures) by hydrogen bonds between base pairs (`A`, `U`), (`C`, `G`) and (`G`, `U`). Therefore, we can regard RNA 2nd-structures as SA-strings. Here, the important fact to note is that a typical type of RNA 2nd-structures called *pseudoknot* forms the crossing dependency(See Fig.1). The existence of pseudoknots leads us to an application of TAGs to modeling RNA 2nd-structures.

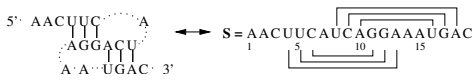


Figure 1: (a): *pseudoknot*. (b): its interpretation as an SA-string.

In this study, we will focus on modeling RNA 2nd-structures as derivations of TAGs and extracting consensus structures by aligning them. Our recent study intends to model the resulting alignment by TAGs. If we adopt conventional alignment methods [2, 5], then we may fail in modeling the resulting alignment by TAGs. Therefore, we will formalize an alignment problem for modeling given RNA 2nd-structures by TAGs. However, there exist a number of derivations corresponding to an RNA 2nd-structure. Then, the question arises of which derivation we should choose for each 2nd-structure. This is the structural ambiguity problem of TAGs. For dealing with this problem appropriately, we will introduce a notion of edit distance between SA-strings which deals with the structural ambiguity problem, and then present a pairwise alignment algorithm for two SA-strings based on the edit distance.

## 2 Tree Adjoining Grammars

For $p, q \in \mathbb{N}^*$, we write $p \leqslant q$ iff there exists $r \in \mathbb{N}^*$ s.t. $p \cdot r = q$, and $p < q$ iff $p \leqslant q$ and $p \neq q$.

Let $V$ be a finite alphabet, and $\Sigma \subset V$. $\Sigma$ is

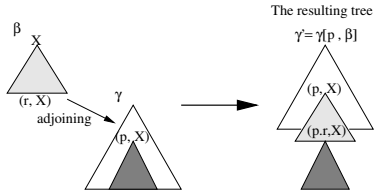


Figure 2: Tree adjoining operation

called a *terminal* alphabet, while $V - \Sigma$ is called a *nonterminal* one. $V^*$ denotes the set of all finite strings over $V$, and $\lambda$ denotes the empty string. Given a string $S$, $S[i]$ denotes the $i$-th symbol of $S$, and $S[i..j]$ denotes the substring from $S[i]$ to $S[j]$ if $i \leq j$ , otherwise $S[i..j] = \lambda$. The length of $S$ is denoted by $|S|$. The concatenation of $n$ substrings $S[x_1..y_1], S[x_2..y_2], \cdots, S[x_n..y_n]$ of $S$ is denoted by $S[x_1..y_1; x_2..y_2; \cdots; x_n..y_n]$, where $y_i < x_{i+1} (1 \leq i \leq n-1)$. We abbreviate the notation to $S[y_1; x_2..y_2; \cdots; x_n]$ when $x_1 = 1$ and $y_n = |S|$.

A *tree* over $V$ is a function $\gamma : \Delta_\gamma \to V$, where $\Delta_\gamma$ is a finite subset of $\mathbb{N}^*$ s.t. (1) if $q \in \Delta_\gamma$ and $p < q$, then $p \in \Delta_\gamma$; (2) if $p{\cdot}j \in \Delta_\gamma$, then $p{\cdot}1, \ldots, p{\cdot}(j-1) \in \Delta_\gamma$. By $\tau_V$, we denote the set of all trees over $V$.

The *yield* is a function $Y : \tau_V \to V^*$ defined recursively as follows:

$$\begin{aligned} Y(\gamma) &= \gamma(0), \text{ if } \Delta_\gamma = \{0\}; \\ Y(\gamma) &= Y(\gamma/1)Y(\gamma/2)\cdots Y(\gamma/j), \\ &\quad \text{if } 1, 2, \ldots, j \in \Delta_\gamma \text{ and } j+1 \notin \Delta_\gamma. \end{aligned}$$

A *tree adjoining grammar* (TAG) $G$ over $(V, \Sigma)$ is defined as a pair $G = (\mathscr{I}, \mathscr{A})$. $\mathscr{I}$ is called the set of *initial trees* of $G$, and $\mathscr{A}$ the set of *adjunct trees* of $G$. $\mathscr{I} \cup \mathscr{A}$ is called the set of *elementary trees* of $G$. The leaf of an adjunct tree $\beta \in \mathscr{A}$ with the label of root is called the *foot node* of $\beta$.

Let $\gamma \in \tau_V, p \in \Delta_\gamma$ and $\beta \in \mathscr{A}$. If $\gamma(p) = \beta(0)$ and NA constraint is not associated with $p$, then $\beta$ *is adjoinable to* $\gamma$ *at* $p$, and the tree obtained by adjoining $\beta$ to $\gamma$ at $p$, denoted by $\gamma[p, \beta]$, is defined as $\gamma[p, \beta] = \gamma \backslash p \cup p \cdot \beta \cup (p \cdot r) \cdot (\gamma/p)$, (See Fig. 2), where $r$ is the address of the foot node of $\beta$, and NA constraint is as follows:

**Null Adjoining (NA):** For a node $n$ in elementary trees, no tree in $\mathscr{A}$ can be adjoined at $n$.

A node is *inactive* iff NA constraint is associated with the node, and *active* otherwise.

An elementary tree is *simple linear* iff all but one node in the tree are inactive. A TAG is *simple linear* iff all elementary trees are simple linear. Especially, a simple linear TAG(SLTAG) $G$ over $(V, \Sigma)$ is said to be *universal* iff $|V - \Sigma| = 1$. Universal SLTAGs prove useful in modeling RNA 2nd-structures [1, 4].

Let $G = (\mathscr{I}, \mathscr{A})$ be an SLTAG. For trees $\gamma, \gamma' \in \tau_V$, we write $\gamma \vdash^\beta \gamma'$ iff there exists $\beta \in \mathscr{A}$ s.t. $\beta$ is adjoinable to $\gamma$ at a unique node of $\gamma$, and $\gamma' = \gamma[\beta]$.

Here, we consider derivation processes of $\gamma_f \in \tau_V$ in $G$. Suppose that there exist an initial tree $\gamma_0 \in \mathscr{I}$, trees $\gamma_1, \ldots, \gamma_n$ and adjunct trees $\beta_1, \ldots, \beta_n$ s.t.

$$\gamma_0 \vdash^{\beta_1} \gamma_1 \vdash^{\beta_2} \gamma_2 \vdash^{\beta_3} \cdots \vdash^{\beta_n} \gamma_n = \gamma_f. \qquad (1)$$

We call a sequence $\beta_1 \cdots \beta_n$ *$\alpha$-derivation of $\gamma_f$ in $G$*. By $\alpha(\gamma, G)$, we denote the set of $\alpha$-derivations of $\gamma$ in $G$. We define the set $\tau(G)$ of derived trees in $G$ as $\tau(G) = \{\gamma \in \tau_V \mid |\alpha(\gamma, G)| \geq 1\}$. Then, we define the set $\alpha(G)$ of $\alpha$-derivations in $G$ as $\alpha(G) = \bigcup_{\gamma \in \tau(G)} \alpha(\gamma, G)$.

Let $S$ be a string over $\Sigma$. We consider a partition $C$ of the set of integers from 1 to $|S|$. We call the string $S$ with its partition $C$ *structure-annotated string* ($SA$-string), denoted by $(S, C)$.

For $\gamma \in \tau(G)$, we say that an $\alpha$-derivation $D_\alpha$ of $\gamma$ *corresponds to* $(S, C)$ iff (1) $Y(\gamma) = S$ and (2) $S[i_1], \ldots, S[i_k]$ are derived by some adjunct tree in $D_\alpha$ at a time iff $\{i_1, \ldots, i_k\} \in C$. By $\alpha((S, C), G)$, we denote a set of $\alpha$-derivations in $\alpha(G)$ which correspond to $(S, C)$.

**[Structural Ambiguity Problem of TAGs]**

Let $G$ be a TAG. We say that G is *structurally ambiguous* iff $\exists (S, C), |\alpha((S, C), G)| > 1$.

One of the causes of structural ambiguity is the existence of more than one active node in a tree at which some adjunct tree is adjoinable. We consider only SLTAGs so that we can not find this kind of structural ambiguity, but SLTAGs still have the kind of structural ambiguity shown in Fig.3.

# 3 Edit Distance

We first define three kinds of edit operations, *insert*, *delete* and *replace*, on $\alpha$-derivations of universal SLTAGs. Inserting (represented as $\lambda \to b$) is to insert a new adjunct tree into $D_\alpha$. For an adjunct tree in $D_\alpha$, deleting the adjunct tree (as $a \to \lambda$) is to delete it from $D_\alpha$, and replacing the adjunct tree (as $a \to b$) is to replace it with another.

Let $\mu$ be a cost function which assigns a nonnegative real number $\mu(a \to b)$ to each edit operation $a \to b$. We extend $\mu$ to a sequence $ES = e_1, \ldots, e_n$ of edit operations by letting $\mu(ES) = \sum_{i=1}^{n} \mu(e_i)$.

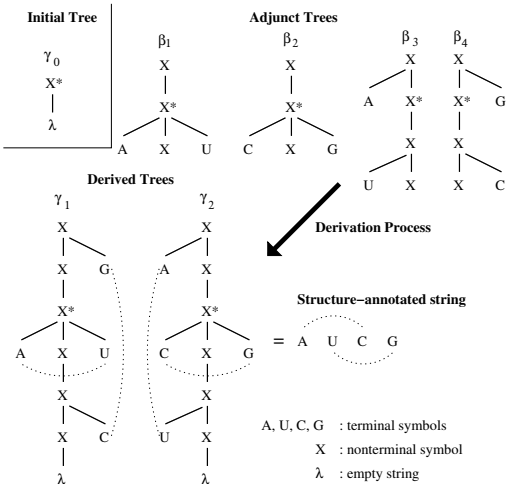


Figure 3: A structural ambiguity of TAGs.

For $\alpha$-derivations $D_\alpha$ and $D'_\alpha$, we define a triplet $(M, D_\alpha, D'_\alpha)$ as a *one-to-one mapping $M$ from $D_\alpha$ to $D'_\alpha$*, where $M$ is a subset of direct product $D_\alpha \times D'_\alpha$ s.t. for any pair of $(D_\alpha[i_1], D'_\alpha[j_1])$ and $(D_\alpha[i_2], D'_\alpha[j_2])$ in $M$, $i_1 < i_2$ iff $j_1 < j_2$.

For a mapping $M$, by $dom(M)$ and $rng(M)$, we denote the *domain* of $M$ and the *range* of $M$, respectively. Then we define the cost of $M$ as

$$\mu((M, D_\alpha, D'_\alpha)) = \sum_{a \notin dom(M)} \mu(a \to \lambda) + \sum_{b \notin rng(M)} \mu(\lambda \to b) + \sum_{(a,b) \in M} \mu(a \to b)$$

**Lemma 1.** For any sequence $ES$ of edit operations transforming $D_\alpha$ to $D'_\alpha$, there exists a mapping $(M, D_\alpha, D'_\alpha)$ s.t. $\mu((M, D_\alpha, D'_\alpha)) \leq \mu(ES)$.

Then, based on Lemma 1, we define an *edit distance between two $\alpha$-derivations* as the minimum cost of mapping from one to the other. Formally, the edit distance between two $\alpha$-derivations $D_\alpha$ and $D'_\alpha$ is defined as follows:

$$D(D_\alpha, D'_\alpha) = \min_M \{\mu((M, D_\alpha, D'_\alpha))\}.$$

Finally, we introduce the notion of an *edit distance between two SA-strings*. Note that it essentially deals with the structural ambiguity problem of SLTAGs. An edit distance between two SA-strings $(S, C)$ and $(S', C')$ is defined as follows:

$$D((S,C),(S',C'),G) = \min_{D_\alpha, D'_\alpha} \{D(D_\alpha, D'_\alpha) \mid D_\alpha \in \alpha((S,C),G) \text{ and } D'_\alpha \in \alpha((S',C'),G)\},$$

# 4 The Alignment Problem for SA-strings

## 4.1 Important properties of SLTAGs

Let $\beta$ be a simple linear adjunct tree with active node at $p$ where $Y(\beta) = a_1 \cdots a_i X a_{i+1} \cdots a_j$ and $\beta(0) = X$. Here note that there must exist indices $i'$ and $j'$ s.t. $Y(\beta/p) = a_{i'} \cdots a_i X a_{i+1} \cdots a_{j'}$ ($1 \leq i' \leq i$, $i+1 \leq j' \leq j$). We define the four subsequences of $Y(\beta)$ by $LU(\beta) = a_1 \cdots a_{i'-1}$, $LD(\beta) = a_{i'} \cdots a_i$, $RD(\beta) = a_{i+1} \cdots a_{j'}$, and $RU(\beta) = a_{j'+1} \cdots a_j$.

Let $G$ be an SLTAG. For an SA-string $(S, C)$, let $\beta_1 \cdots \beta_n \in \alpha((S, C), G)$ and $\gamma_0 \cdots \gamma_n$ be its corresponding derived trees satisfying formula (1).

**Property 1.** For every $i$ with $0 \leq i \leq n$, there exist four indices $x_i$, $y_i$, $z_i$ and $w_i$ s.t. $0 \leq x_i < y_i$ and $y_i - 1 \leq z_i < w_i \leq |S| + 1$. The quadruple $(x_i, y_i, z_i, w_i)$ is called *factor of $\gamma_i$*, and factorizes $Y(\gamma_i)$ into three substrings of $S$ as follows:

$$Y(\gamma_i) = S[1..x_i]S[y_i..z_i]S[w_i..|S|],$$

where

$$\begin{aligned} S[1..x_i] &= LU(\beta_1) \cdots LU(\beta_i), \\ S[y_i..z_i] &= LD(\beta_i) \cdots LD(\beta_1) RD(\beta_1) \cdots RD(\beta_i), \\ S[w_i..|S|] &= RU(\beta_i) \cdots RU(\beta_1). \end{aligned}$$

For $\beta_1 \cdots \beta_n \in \alpha((S, C), G)$, we call its prefix $\beta_1 \cdots \beta_i$ *$\alpha$-prefix of $(S, C)$*. If an $\alpha$-prefix of $(S, C)$ derives a tree whose factor is $(x, y, z, w)$, then we say that the $\alpha$-prefix *derives* a sub-SA-string $(S, C)[x; y..z; w] = (S[x; y..z; w], C[x; y..z; w])$ of $(S, C)$, where $C[x; y..z; w]$ is a set of structures in $C$ defined on the string $S[x; y..z; w]$.

By $\alpha((S, C)[x; y..z; w], G)$, we denote the set of $\alpha$-prefixes of $(S, C)$ which derive $(S, C)[x; y..z; w]$.

**Proposition 1.** $\lambda \in \alpha((S, C)[x; y..z; w], G)$ iff $x = 0$, $y - 1 = z$, and $w = |S| + 1$ hold.

## 4.2 A DP-algorithm

Let $G$ be a universal SLTAG. For SA-strings $(S, C)$ and $(S', C')$, we describe a DP-algorithm that computes $D((S, C), (S', C'), G)$ in $O(|S|^4 |S'|^4)$ time.

An entry $DP(x, y, z, w; x', y', z', w')$ denotes the minimum value of edit distance between an $\alpha$-prefix of $(S, C)$ which derives $(S, C)[x; y..z; w]$ and an $\alpha$-prefix of $(S', C')$ which derives $(S', C')[x'; y'..z'; w']$. The following recurrence relation would suffice for the development of a DP-algorithm.

**Recurrence 1.** For any $x, y, z, w, x', y', z', w' \in \mathbb{N}$,

**Case 1:** (See Proposition 1) if $x = 0$, $y - 1 = z$, $w = |S| + 1$, $x' = 0$, $y' - 1 = z'$, and $w' = |S'| + 1$, then $DP(x, y, z, w; x', y', z', w') = 0$.

**Case 2:** otherwise,

$$DP(x, y, z, w; x', y', z', w') = \min_{\beta, \beta'} \begin{cases} DP(x - \ell u(\beta), y + \ell d(\beta), \\ \quad z - rd(\beta), w + ru(\beta); \\ \quad x', y', z', w') + \mu(\beta \to \lambda), \\ \\ DP(x, y, z, w; x' - \ell u(\beta'), y' + \ell d(\beta'), \\ \quad z' - rd(\beta'), w' + ru(\beta')) \\ \quad +\mu(\lambda \to \beta'), \\ \\ DP(x - \ell u(\beta), y + \ell d(\beta), \\ \quad z - rd(\beta), w + ru(\beta); \\ \quad x' - \ell u(\beta'), y' + \ell d(\beta'), \\ \quad z' - rd(\beta'), w' + ru(\beta')) \\ \quad +\mu(\beta \to \beta'), \\ \\ \infty, \end{cases}$$

where $\beta$ and $\beta'$ are adjunct trees satisfying the following condition:

**Condition 1.**

1. The subsequence of $S$ corresponding to $Y(\beta)$ forms a structure in $C[x; y..z; w]$.
2. $LU(\beta) = S[x - \ell u(\beta) + 1 \,..\, x]$,
3. $LD(\beta) = S[y \,..\, y + \ell d(\beta) - 1]$,
4. $RD(\beta) = S[z - rd(\beta) + 1 \,..\, z]$,
5. $RU(\beta) = S[w \,..\, w + ru(\beta) - 1]$.

The same condition must apply to $\beta'$, $x'$, $y'$, $z'$, $w'$, $S'$, and $C'[x'; y'..z'; w']$.

**Lemma 2.** In Case 2, all candidates of $\beta$ and $\beta'$ can be enumerated in $O(\xi\xi'|\mathscr{A}|^2)$ time, where $\xi$ is the maximum cardinality of structures in $C$, and $\xi'$ is the maximum cardinality of structures in $C'$.

Based on Recurrence 1, one can easily design a DP-algorithm to compute the edit distance between $(S, C)$ and $(S', C')$ as follows:

```
1. procedure CONSTRUCT(DP, G)
   begin
     for x = 0 to |S| do
      for w = |S| + 1 downto x + 1 do
        for z = x + 1 to w − 1 do
         for y = z + 1 downto x + 1 do
           for x' = 0 to |S'| do
            for w' = |S'| + 1 downto x' + 1 do
              for z' = x' + 1 to w' − 1 do
               for y' = z' + 1 downto x' + 1 do
                 compute DP(x, y, z, w; x', y', z', w')
                   according to Recurrence 1;
   end
```

```
2. procedure DETERMINE(DP)
   begin
     min = ∞;
     for x = 0 to |S| do
      for z = x to |S| do
        for x' = 0 to |S'| do
         for z' = x' to |S'| do
           if DP(x, x+1, z, z+1; x', x'+1, z', z'+1) < min
             min = DP(x, x+1, z, z+1; x', x'+1, z', z'+1);
     return min;
   end
```

```
3. procedure MAIN((S, C), (S', C'), G)
   begin
     matrix DP;
     CONSTRUCT(DP, G);
     output DETERMINE(DP);
   end
```

## 4.3 Correctness and complexity of the algorithm

**Theorem 1.** For two SA-strings $(S, C)$ and $(S', C')$, MAIN$((S, C), (S', C'), G)$ outputs the edit distance between $(S, C)$ and $(S', C')$.

It is obvious from the manner of implementation and Lemma 2 that the time complexity of CONSTRUCT is $O(\xi\xi'|\mathscr{A}|^2|S|^4|S'|^4)$, and that of DETERMINE is $O(|S|^2|S'|^2)$. Therefore the time complexity of MAIN is $O(\xi\xi'|\mathscr{A}|^2|S|^4|S'|^4)$.

# References


[1] H. Asakawa, "Parsing Universal Tree Adjoining Grammars using Structural Information", Master's thesis, Dept. of Comp. Sci., Univ. of Electr.-Communi., 2004. (in Japanese).

[2] T. Jiang, G. Lin, B. Ma, and K. Zhang, "A General Edit Distance between RNA Structures", Proc. 5th Annual International Conference on Computational Molecular Biology (RECOMB'01), pp. 211-220, June 2001.

[3] Y. Uemura, A. Hasegawa, S. Kobayashi, and T. Yokomori, "Tree adjoining grammars for RNA structure prediction", Theoretical Computer Science, vol.210, pp.277-303, 1999.

[4] K. Zhang, L. Wang, and B. Ma, "Computing similarity between RNA structures", Proc. 10th Annual Symposium on Combinatorial Pattern Matching(CPM'99), LNCS 1645, Springer, pp.281-293, 1999.