*Original Paper*

# RNA Pseudoknotted Structure Prediction Using Stochastic Multiple Context-Free Grammar

Yuki Kato,† Hiroyuki Seki† and Tadao Kasami†

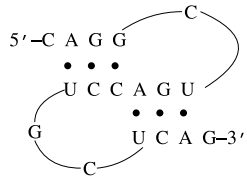
Many attempts have so far been made at modeling RNA secondary structure by formal grammars. In a grammatical approach, secondary structure prediction can be viewed as parsing problem. However, there may be many different derivation trees for an input sequence. Thus, it is necessary to have a method of extracting biologically realistic derivation trees among them. One solution to this problem is to extend a grammar to a probabilistic model and find the most likely derivation tree, and another is to take free energy minimization into account. One simple formalism for describing RNA folding is context-free grammars (CFGs), but it is known that CFGs cannot represent pseudoknots. Therefore, several formal grammars have been proposed for modeling RNA pseudoknotted structure. In this paper, we focus on multiple context-free grammars (MCFGs), which are natural extension of CFGs and can represent pseudoknots, and extend MCFGs to a probabilistic model called stochastic MCFG (SMCFG). We present a polynomial time parsing algorithm for finding the most probable derivation tree, which is applicable to RNA secondary structure prediction including pseudoknots. Also, we propose a probability parameter estimation algorithm based on the EM (expectation maximization) algorithm. Finally, we show some experimental results on RNA pseudoknot prediction using the SMCFG parsing algorithm, which show good prediction accuracy.

## 1. Introduction

Non-coding RNAs fold into characteristic structures determined by interactions between mostly Watson-Crick complementary base pairs. Such a base paired structure is called the *secondary structure*. *Pseudoknot* (**Fig. 1** (a)) is one of the typical substructures found in the secondary structures of several RNAs, including rRNAs, tmRNAs and viral RNAs. An alternative graphic representation of a pseudoknot is arc depiction where arcs connect base pairs (Fig. 1 (b)). It has been recognized that pseudoknots play an important role in RNA functions such as ribosomal frameshifting and regulation of translation.

Many attempts have so far been made at modeling RNA secondary structure by formal grammars. In a grammatical approach, secondary structure prediction can be viewed as parsing problem. However, there may be many different derivation trees for an input sequence. Thus, it is necessary to have a method of extracting biologically realistic derivation trees among them. One solution to this problem is to extend a grammar to a probabilistic model and find the most likely derivation tree, and another is to take free energy minimization into account. Eddy and Durbin [5], and Sakakibara, et al. [13] modeled RNA secondary structure without pseudoknots by using stochastic context-free grammars (stochastic CFGs or SCFGs). For pseudoknotted structure, however, another approach has to be taken since a single CFG cannot represent crossing dependencies of base pairs in pseudoknots (Fig. 1 (b)) for the lack of generative power. Brown and Wilson [2] proposed a model based on intersections of SCFGs to describe RNA pseudoknots. Cai, et al. [3] introduced a model based on parallel communication grammar systems using a single CFG synchronized with a number of regular grammars. Akutsu [1] provided dynamic programming algorithms for RNA pseudoknot prediction without using grammars. On the other hand, several grammars have been proposed where the grammar itself can fully describe pseudoknots. Rivas and Eddy [11),12] provided a dynamic programming algorithm for predicting RNA secondary structure including pseudoknots, and introduced a new class of grammars called RNA pseudoknot grammars (RPGs) for deriving sequences with gap. Uemura, et al. [15] defined specific subclasses of tree adjoining grammars (TAGs) named SLTAGs and extended SLTAGs (ESLTAGs) respectively, and predicted RNA pseudoknots by using parsing algorithm

† Graduate School of Information Science, Nara Institute of Science and Technology




(a) Pseudoknot


(b) Arc depiction of (a)

**Fig. 1** Example of RNA secondary structure.

of ESLTAG. Matsui, et al. [10] proposed pair stochastic tree adjoining grammars (PSTAGs) based on ESLTAGs and tree automata for aligning and predicting pseudoknots, which showed good prediction accuracy. These grammars have generative power stronger than CFGs and polynomial time algorithms for parsing problem.

In our previous work [8], we have identified RPGs, SLTAGs and ESLTAGs as subclasses of *multiple context-free grammars* (MCFGs) [7),14)], which can model RNA pseudoknots, and have shown a candidate subclass of the minimum grammars for representing pseudoknots. The generative power of MCFGs is stronger than that of CFGs and MCFGs have a polynomial time parsing algorithm like the CYK (Cocke-Younger-Kasami) algorithm for CFGs.

In this paper, we extend MCFGs to a probabilistic model called stochastic MCFG (SMCFG). We present a polynomial time parsing algorithm for finding the most probable derivation tree, which is applicable to RNA secondary structure prediction including pseudoknots. Also, we propose a probability parameter estimation algorithm based on the EM (expectation maximization) algorithm. Finally, we show some experimental results on pseudoknot prediction for three RNA families using the SMCFG parsing algorithm, which show good prediction accuracy.

## 2. Stochastic Multiple Context-Free Grammar

For an alphabet $\Sigma$, let $\Sigma^*$ denote the set of all finite sequences over $\Sigma$. The empty sequence is denoted by $\varepsilon$. For a sequence $w \in \Sigma^*$, let $|w|$ denote the length of $w$, that is, the number of symbols occurring in $w$.

A *stochastic multiple context-free grammar* (stochastic MCFG, or SMCFG) is a probabilistic extension of MCFG [7),14)]. An SMCFG is a 5-tuple $G = (N, T, F, P, S)$ where $N$ is a finite set of nonterminals, $T$ is a finite set of terminals, $F$ is a finite set of functions, $P$ is a finite set of (production) rules and $S \in N$ is the start symbol. For each $A \in N$, a positive integer denoted by $\dim(A)$ is given and $A$ derives $\dim(A)$-tuples of terminal sequences. For the start symbol $S$, $\dim(S) = 1$. For each $f \in F$, positive integers $d_i$ $(0 \le i \le k)$ are given and $f$ is a total function from $(T^*)^{d_1} \times \cdots \times (T^*)^{d_k}$ to $(T^*)^{d_0}$ satisfying the following condition (F):

**(F)** Let $\overline{x_i} = (x_{i1}, \ldots, x_{id_i})$ denote the $i$th argument of $f$ for $1 \le i \le k$. The $h$th component of the function value for $1 \le h \le d_0$, denoted by $f^{[h]}$, is defined as

$$f^{[h]}[\overline{x_1}, \ldots, \overline{x_k}] = \beta_{h0} z_{h1} \beta_{h1} z_{h2} \cdots z_{hv_h} \beta_{hv_h} \quad (1)$$

where $\beta_{hl} \in T^*$ $(0 \le l \le v_h)$ and $z_{hl} \in \{x_{ij} \mid 1 \le i \le k,\ 1 \le j \le d_i\}$ $(1 \le l \le v_h)$. The total number of occurrences of $x_{ij}$ in the right-hand sides of (1) from $h = 1$ through $d_0$ is at most one.

Each rule in $P$ has the form of $A_0 \xrightarrow{p} f[A_1, \ldots, A_k]$ where $A_i \in N$ $(0 \le i \le k)$, $f : (T^*)^{\dim(A_1)} \times \cdots \times (T^*)^{\dim(A_k)} \to (T^*)^{\dim(A_0)} \in F$ and $p$ is a real number with $0 < p \le 1$ called the *probability* of this rule. The summation of the probabilities of the rules with the same left-hand side should be one. If we are not interested in $p$, we just write $A_0 \to f[A_1, \ldots, A_k]$. If $k \ge 1$, the rule is called a *nonterminating rule*, and if $k = 0$, it is called a *terminating rule*. A terminating rule $A_0 \to f[\,]$ with $f^{[h]}[\,] = \beta_h$ $(1 \le h \le \dim(A_0))$ is simply written as $A_0 \to (\beta_1, \ldots, \beta_{\dim(A_0)})$.

We define derivation trees as follows:

**(D1)** If $A \xrightarrow{p} \overline{\alpha} \in P$ $(\overline{\alpha} \in (T^*)^{\dim(A)})$, then the ordered tree with the root labeled $A$ which has $\overline{\alpha}$ as the only one child is a derivation tree for $\overline{\alpha}$ with probability $p$.

**(D2)** If $A \xrightarrow{p} f[A_1, \ldots, A_k] \in P$ and $t_1, \ldots, t_k$ with the roots labeled $A_1, \ldots, A_k$ are derivation trees for $\overline{\alpha_1}, \ldots, \overline{\alpha_k}$ with probabilities $p_1, \ldots, p_k$, respectively, then the ordered tree with the root labeled $A$ (or $A : f$ if necessary) which has $t_1, \ldots, t_k$ as (immediate) subtrees from left to right is a derivation tree for $f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ with probability $p \cdot \prod_{i=1}^{k} p_i$.

For $A \in N$, $\overline{\alpha} \in (T^*)^{\dim(A)}$ and $q$ $(0 < q \le 1)$, we write $A \overset{*}{\Rightarrow} \overline{\alpha}$ with probability $q$

if $q$ is the summation of the probabilities of derivation trees for $\overline{\alpha}$ with the root labeled $A$. The language generated by an SMCFG $G$ is defined as $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w$ with probability greater than $0\}$.

**Example 1.** Let $G_1 = (N_1, T_1, F_1, P_1, S)$ be an SMCFG where $N_1 = \{S, A\}$, $T_1 = \{a, b\}$ and $P_1 = \{S \overset{1}{\to} J[A],\ A \overset{0.3}{\to} f[A],\ A \overset{0.7}{\to} (ab, cd)\}$ where $\dim(S) = 1$, $\dim(A) = 2$, $J[(x_1, x_2)] = x_1x_2$ and $f[(x_1, x_2)] = (ax_1b, cx_2d)$. Then, $A \overset{*}{\Rightarrow} (ab, cd)$ with probability 0.7 by the third rule, which is followed by $A \overset{*}{\Rightarrow} f[(ab, cd)] = (aabb, ccdd)$ with probability $0.3 \cdot 0.7 = 0.21$ by the second rule. Also, by the first rule, $S \overset{*}{\Rightarrow} J[(aabb, ccdd)] = aabbccdd$ with probability $1 \cdot 0.21 = 0.21$. In fact, $L(G_1) = \{a^n b^n c^n d^n \mid n \geq 1\}$. □

**Example 2.** Consider an MCFG $G_2 = (\{S, A, X_1, X_2\}, \{a, c, g, u\}, F_2, P_2, S)$ ☆ for generating RNA sequences where $P_2$ and $F_2$ are as follows:

$S \to J[A]$,
$A \to UP^{\alpha}_{1L}[A],\ X_1 \to UP^{\alpha}_{1L}[A]$,
$A \to UP^{\alpha}_{1R}[A],\ A \to UP^{\alpha}_{1R}[X_1]$,
$A \to UP^{\alpha}_{2L}[A],\ X_2 \to UP^{\alpha}_{2L}[A]$,
$A \to UP^{\alpha}_{2R}[A],\ A \to UP^{\alpha}_{2R}[X_2]$,
$A \to BP^{\alpha\beta}[A]$,
$A \to (\varepsilon, \varepsilon)$,
$J[(x_1, x_2)] = x_1x_2$,
$UP^{\alpha}_{1L}[(x_1, x_2)] = (\alpha x_1, x_2)$,
$UP^{\alpha}_{1R}[(x_1, x_2)] = (x_1\alpha, x_2)$,
$UP^{\alpha}_{2L}[(x_1, x_2)] = (x_1, \alpha x_2)$,
$UP^{\alpha}_{2R}[(x_1, x_2)] = (x_1, x_2\alpha)$,
$BP^{\alpha\beta}[(x_1, x_2)] = (\alpha x_1, x_2\beta)$.

Note that $\alpha \in \{a, c, g, u\}$ and $(\alpha, \beta) \in \{(a, u), (u, a), (c, g), (g, c)\}$. Functions have mnemonic names where $UP$ and $BP$ stand for unpair and base pair respectively. The RNA sequence `agacuu` in **Fig. 2** can be generated by the above rules as follows: $A \overset{*}{\Rightarrow} BP^{gc}[(\varepsilon, \varepsilon)] = (g, c)$, $A \overset{*}{\Rightarrow} BP^{au}[(g, c)] = (ag, cu)$, $X_2 \overset{*}{\Rightarrow} UP^{a}_{2L}[(ag, cu)] = (ag, acu)$, $A \overset{*}{\Rightarrow} UP^{u}_{2R}[(ag, acu)] = (ag, acuu)$ and $S \overset{*}{\Rightarrow} J[(ag, acuu)] = agacuu$. $G_2$ has a derivation tree (**Fig. 3**) for `agacuu` which represents the pseudoknot shown in Fig. 2. □

In this paper, we focus on an SMCFG $G_R = (N, T, F, P, S)$ that satisfies the following conditions: $G_R$ has $m$ different nonterminals denoted by $W_1, \ldots, W_m$, each of which uses the only one type of a rule denoted by E, S, D, $B_1$, $B_2$, $B_3$, $B_4$, $U_{1L}$, $U_{1R}$, $U_{2L}$, $U_{2R}$ or P ☆☆ (see **Table 1**). The type of $W_v$ is denoted by type($v$) and we predefine type(1) = S, that is, $W_1$ is the start symbol. Consider a sample rule set $W_v \to UP^{\alpha}_{1L}[W_y] \mid UP^{\alpha}_{1L}[W_z]$ where $UP^{\alpha}_{1L}[(x_1, x_2)] = (\alpha x_1, x_2)$ and $\alpha \in T$. For each rule $r$, two real values called *transition probability* $p_1$ and *emission probability* $p_2$ are specified as shown in Table 1. The probability of $r$ is simply defined as $p_1 \cdot p_2$. In application, $p_1 = t_v(y)$ and $p_2 = e_v(a_i), \ldots$ in Table 1 are parameters for the grammar, which are set by hand or by a training algorithm (Section 3.3) depending on the set of possible sequences to be analyzed. $G_R$ can generate RNA sequences corresponding to pseudoknots (see Example 2 and Ref. 9)).

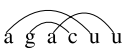



**Fig. 2** Example of a pseudoknot.

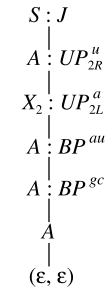



**Fig. 3** A derivation tree in $G_2$.

## 3. Algorithms for SMCFG

In RNA structure analysis using stochastic grammars, we have to deal with the following three problems [4]:

(1) Calculate the optimal alignment of a sequence to a stochastic grammar. (alignment problem)
(2) Calculate the probability of a sequence, given a stochastic grammar. (scoring problem)
(3) Estimate optimal probability parameters for a stochastic grammar, given a set of example sequences. (training problem)

In this section, we give solutions to each

☆ For simplicity, we consider a non-stochastic grammar here.

☆☆ These types stand for End, Start, Delete, Bifurcation, Unpair and Pair respectively.

**Table 1** SMCFG $G_R$.

| Type | Rule set | Function | Transition prob. | Emission prob. |
|---|---|---|---|---|
| E | $W_v \to (\varepsilon, \varepsilon)$ | | 1 | 1 |
| S | $W_v \to J[W_y]$ | $J[(x_1, x_2)] = x_1 x_2$ | $t_v(y)$ | 1 |
| D | $W_v \to SK[W_y]$ | $SK[(x_1, x_2)] = (x_1, x_2)$ | $t_v(y)$ | 1 |
| $B_1$ | $W_v \to C_1[W_y, W_z]$ | $C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22})$ | 1 | 1 |
| $B_2$ | $W_v \to C_2[W_y, W_z]$ | $C_2[x_1, (x_{21}, x_{22})] = (x_{21} x_1, x_{22})$ | 1 | 1 |
| $B_3$ | $W_v \to C_3[W_y, W_z]$ | $C_3[x_1, (x_{21}, x_{22})] = (x_{21}, x_1 x_{22})$ | 1 | 1 |
| $B_4$ | $W_v \to C_4[W_y, W_z]$ | $C_4[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22} x_1)$ | 1 | 1 |
| $U_{1L}$ | $W_v \to UP_{1L}^{a_i}[W_y]$ | $UP_{1L}^{a_i}[(x_1, x_2)] = (a_i x_1, x_2)$ | $t_v(y)$ | $e_v(a_i)$ |
| $U_{1R}$ | $W_v \to UP_{1R}^{a_j}[W_y]$ | $UP_{1R}^{a_j}[(x_1, x_2)] = (x_1 a_j, x_2)$ | $t_v(y)$ | $e_v(a_j)$ |
| $U_{2L}$ | $W_v \to UP_{2L}^{a_k}[W_y]$ | $UP_{2L}^{a_k}[(x_1, x_2)] = (x_1, a_k x_2)$ | $t_v(y)$ | $e_v(a_k)$ |
| $U_{2R}$ | $W_v \to UP_{2R}^{a_l}[W_y]$ | $UP_{2R}^{a_l}[(x_1, x_2)] = (x_1, x_2 a_l)$ | $t_v(y)$ | $e_v(a_l)$ |
| P | $W_v \to BP^{a_i a_l}[W_y]$ | $BP^{a_i a_l}[(x_1, x_2)] = (a_i x_1, x_2 a_l)$ | $t_v(y)$ | $e_v(a_i, a_l)$ |

problem for the specific SMCFG $G_R = (N, T, F, P, S)$.

### 3.1 Alignment Problem

The alignment problem for $G_R$ is to find the most probable derivation tree for a given input sequence. This problem can be solved by a dynamic programming algorithm similar to the CYK algorithm for SCFGs [4], and in this paper, we also call the parsing algorithm for $G_R$ the CYK algorithm. We fix an input sequence $w = a_1 \cdots a_n$ ($|w| = n$). In fact, $w$ is an RNA sequence composed of four symbols $a$, $c$, $g$ and $u$. Let $\gamma_v(i, j)$ and $\gamma_y(i, j, k, l)$ be the logarithm of maximum probabilities of a derivation subtree rooted at a nonterminal $W_v$ for a terminal subsequence $a_i \cdots a_j$ and of a derivation subtree rooted at a nonterminal $W_y$ for a pair of terminal subsequences $(a_i \cdots a_j, a_k \cdots a_l)$ respectively. The variables $\gamma_v(i, i-1)$ and $\gamma_y(i, i-1, k, k-1)$ are the logarithm of maximum probabilities for an empty sequence $\varepsilon$ and a pair of $\varepsilon$. Let $\tau_v(i, j)$ and $\tau_y(i, j, k, l)$ be traceback variables for constructing a derivation tree, which are calculated together with $\gamma_v(i, j)$ and $\gamma_y(i, j, k, l)$. We define $\mathcal{C}_v = \{y \mid W_v \to f[W_y] \in P,\ f \in F\}$. To avoid non-emitting cycles, we assume that the nonterminals are numbered such that $v < y$ for all $y \in \mathcal{C}_v$. The CYK algorithm uses a five dimensional dynamic programming matrix to calculate $\gamma$, which leads to $\log P(w, \hat{\pi} \mid \theta)$ where $\hat{\pi}$ is the most probable derivation tree and $\theta$ is an entire set of probability parameters. The illustration of the iteration step in the CYK algorithm is shown in **Fig. 4**. The detailed description of the algorithm is as follows:

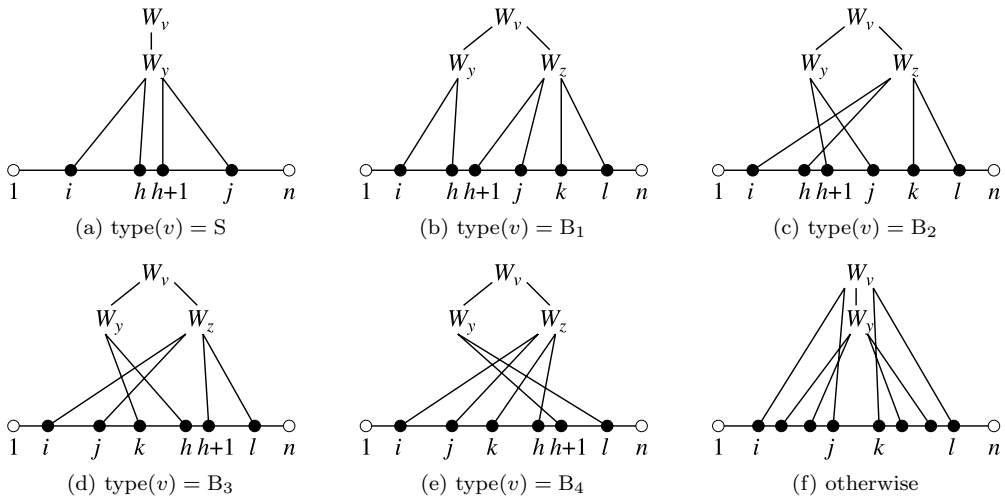
**Algorithm 1** (CYK).

**Initialization:**

**for** $i \leftarrow 1$ **to** $n+1$, $k \leftarrow i$ **to** $n+1$, $v \leftarrow 1$ **to** $m$
  **do if** type$(v) =$ E
    **then** $\gamma_v(i, i-1, k, k-1) \leftarrow 0$
  **else** $\gamma_v(i, i-1, k, k-1) \leftarrow -\infty$

**Iteration:**

**for** $i \leftarrow n$ **downto** 1, $j \leftarrow i-1$ **to** $n$, $k \leftarrow n+1$ **downto** $j+1$, $l \leftarrow k-1$ **to** $n$, $v \leftarrow 1$ **to** $m$
  **do if** type$(v) =$ E
    **then if** $j = i-1$ **and** $l = k-1$
      **then** skip
    **else** $\gamma_v(i, j, k, l) \leftarrow -\infty$
  **if** type$(v) =$ S
    **then** $\gamma_v(i, j)$
      $\leftarrow \max_{y \in \mathcal{C}_v} \max_{h=i-1,\ldots,j} [\log t_v(y) + \gamma_y(i, h, h+1, j)]$
      $\tau_v(i, j)$
      $\leftarrow \arg\max_{(y,h)} [\log t_v(y) + \gamma_y(i, h, h+1, j)]$
  **if** type$(v) =$ $B_1$ **and** $W_v \to C_1[W_y, W_z]$
    **then** $\gamma_v(i, j, k, l)$
      $\leftarrow \max_{h=i-1,\ldots,j} [\gamma_y(i, h) + \gamma_z(h+1, j, k, l)]$
      $\tau_v(i, j, k, l)$
      $\leftarrow \arg\max_{(y,z,h)} [\gamma_y(i, h) + \gamma_z(h+1, j, k, l)]$
  **if** type$(v) =$ $B_2$ **and** $W_v \to C_2[W_y, W_z]$
    **then** $\gamma_v(i, j, k, l)$
      $\leftarrow \max_{h=i-1,\ldots,j} [\gamma_y(h+1, j) + \gamma_z(i, h, k, l)]$
      $\tau_v(i, j, k, l)$
      $\leftarrow \arg\max_{(y,z,h)} [\gamma_y(h+1, j) + \gamma_z(i, h, k, l)]$
  **if** type$(v) =$ $B_3$ **and** $W_v \to C_3[W_y, W_z]$
    **then** $\gamma_v(i, j, k, l)$
      $\leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i, j, h+1, l) + \gamma_y(k, h)]$
      $\tau_v(i, j, k, l)$
      $\leftarrow \arg\max_{(y,z,h)} [\gamma_z(i, j, h+1, l) + \gamma_y(k, h)]$
  **if** type$(v) =$ $B_4$ **and** $W_v \to C_4[W_y, W_z]$
    **then** $\gamma_v(i, j, k, l)$




(a) type($v$) = S (b) type($v$) = $B_1$ (c) type($v$) = $B_2$

(d) type($v$) = $B_3$ (e) type($v$) = $B_4$ (f) otherwise

**Fig. 4** Illustration of the iteration step for calculating $\gamma$.

$$\leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i,j,k,h)+\gamma_y(h+1,l)]$$
$$\tau_v(i,j,k,l)$$
$$\leftarrow \arg\max_{(y,z,h)} [\gamma_z(i,j,k,h)+\gamma_y(h+1,l)]$$

**if** type($v$) = P
  **then if** $j = i-1$ **or** $l = k-1$
    **then** $\gamma_v(i,j,k,l) \leftarrow -\infty$
  **else** $\gamma_v(i,j,k,l)$
$$\leftarrow \max_{y\in\mathcal{C}_v}[\log e_v(a_i,a_l) + \log t_v(y) + \gamma_y(i+1,j,k,l-1)]$$
$$\tau_v(i,j,k,l)$$
$$\leftarrow \arg\max_{y}[\log e_v(a_i,a_l) + \log t_v(y) + \gamma_y(i+1,j,k,l-1)]$$
**else** $\gamma_v(i,j,k,l)$
$$\leftarrow \max_{y\in\mathcal{C}_v}[\log e_v(a_i,a_j,a_k,a_l) + \log t_v(y) + \gamma_y(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})]$$
$$\tau_v(i,j,k,l)$$
$$\leftarrow \arg\max_{y}[\log e_v(a_i,a_j,a_k,a_l) + \log t_v(y) + \gamma_y(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})]$$

Note: $e_v(a_i,a_j,a_k,a_l) = e_v(a_i)$ for type($v$) = $U_{1L}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_j)$ for type($v$) = $U_{1R}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_k)$ for type($v$) = $U_{2L}$, $e_v(a_i,a_j,a_k,a_l) = e_v(a_l)$ for type($v$) = $U_{2R}$, $e_v(a_i,a_j,a_k,a_l) = 1$ for the other types except P. Also, $\Delta_v^{1L} = 1$ for type($v$) = $U_{1L}$, $\Delta_v^{1R} = 1$ for type($v$) = $U_{1R}$, $\Delta_v^{2L} = 1$ for type($v$) = $U_{2L}$, $\Delta_v^{2R} = 1$ for type($v$) = $U_{2R}$, and $\Delta_v^{1L},\ldots,\Delta_v^{2R}$ are set to 0 for the other types except P. □

When the calculation terminates, we obtain $\log P(w,\hat{\pi}\mid\theta) = \gamma_1(1,n)$. If there are $b$ BIFURCATION nonterminals and $a$ other nonterminals, the time and space complexities of the CYK algorithm are $O(amn^4 + bn^5)$ and $O(mn^4)$, respectively. To recover the optimal derivation tree, we use the traceback variables $\tau$ and the push-down stack holding tuples of integers of the forms $(v,i,j)$ and $(y,i,j,k,l)$. The full description of the traceback algorithm is omitted (see Ref. 9)).

### 3.2 Scoring Problem

As in SCFGs [4], the scoring problem for $G_R$ can be solved by the inside algorithm. The inside algorithm calculates the summed probabilities $\alpha_v(i,j)$ and $\alpha_y(i,j,k,l)$ of all derivation subtrees rooted at a nonterminal $W_v$ for a subsequence $a_i\cdots a_j$ and of all derivation subtrees rooted at a nonterminal $W_y$ for a pair of subsequences $(a_i\cdots a_j, a_k\cdots a_l)$ respectively. The variables $\alpha_v(i,i-1)$ and $\alpha_y(i,i-1,k,k-1)$ are defined for empty sequences in a similar way to the CYK algorithm. Therefore, we can easily obtain the inside algorithm by replacing max operations with summations in the CYK algorithm (see Ref. 9)). When the calculation terminates, we obtain the probability $P(w\mid\theta) = \alpha_1(1,n)$. The time and space complexities of the algorithm are identical with those of the CYK algorithm.

In order to re-estimate the probability parameters of $G_R$, we need the outside algorithm. The outside algorithm calculates the summed probability $\beta_v(i,j)$ of all derivation trees excluding subtrees rooted at a nonterminal $W_v$ generating a subsequence $a_i\cdots a_j$. Also, it calculates $\beta_y(i,j,k,l)$, the summed probability of all derivation trees excluding subtrees rooted at a nonterminal $W_y$ generating a pair of sub-

sequences $(a_i \cdots a_j, a_k \cdots a_l)$. In the algorithm, we will use $\mathcal{P}_v = \{y \mid W_y \to f[W_v] \in P,\ f \in F\}$. Note that calculating the outside variables $\beta$ requires the inside variables $\alpha$. Unlike CYK and inside algorithms, the outside algorithm recursively works its way inward. The time and space complexities of the outside algorithm are the same as those of CYK and inside algorithms. Formal description of the outside algorithm is shown in Appendix A.1.

### 3.3 Training Problem

The training problem for $G_R$ can be solved by the EM algorithm called the inside-outside algorithm where the inside variables $\alpha$ and outside variables $\beta$ are used to re-estimate probability parameters.

First, we consider the probability that a nonterminal $W_v$ is used at positions $i$, $j$, $k$ and $l$ in a derivation of a single sequence $w$. If type$(v)$ = S, the probability is $\frac{1}{P(w|\theta)}\alpha_v(i,j)\beta_v(i,j)$, otherwise $\frac{1}{P(w|\theta)}\alpha_v(i,j,k,l)\beta_v(i,j,k,l)$. By summing these over all positions in the sequence, we can obtain the expected number of times that $W_v$ is used for $w$ as follows: for type$(v)$ = S, the expected count is

$$\frac{1}{P(w \mid \theta)} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \alpha_v(i,j)\beta_v(i,j),$$

otherwise

$$\frac{1}{P(w \mid \theta)} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \alpha_v(i,j,k,l) \beta_v(i,j,k,l).$$

Next, we extend these expected values from a single sequence $w$ to multiple independent sequences $w^{(r)}$ $(1 \le r \le N)$. Let $\alpha^{(r)}$ and $\beta^{(r)}$ be the inside and outside variables calculated for each input sequence $w^{(r)}$. Then we can obtain the expected number of times $E(v)$ that a nonterminal $W_v$ is used for training sequences $w^{(r)}$ $(1 \le r \le N)$ by summing the above terms over all sequences: for type$(v)$ = S,

$$E(v) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \alpha_v^{(r)}(i,j) \beta_v^{(r)}(i,j),$$

otherwise

$$E(v) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \frac{1}{P(w^{(r)} \mid \theta)} \alpha_v^{(r)}(i,j,k,l)\beta_v^{(r)}(i,j,k,l).$$

Similarly, for a given $W_y$, the expected number of times $E(v \to y)$ that a rule $W_v \to f[W_y]$ is applied can be obtained (see Appendix A.2). For a given terminal $a$ or a pair of terminals $(a,b)$, we can also obtain the expected number of times $E(v \to a)$ (or $E(v \to ab)$) that a rule containing $a$ (or $a$ and $b$) is applied, as shown in Appendix A.2.

Now, we re-estimate probability parameters by using the above expected counts. Let $\hat{t}_v(y)$ be the re-estimated probability that a rule $W_v \to f[W_y]$ is applied. Also, let $\hat{e}_v(a)$ (or $\hat{e}_v(a,b)$) be the re-estimated probability that a rule containing $a$ (or $a$ and $b$) is applied. We can obtain each re-estimated probability by the following equations:

$$\hat{t}_v(y) = \frac{E(v \to y)}{E(v)},\ \hat{e}_v(a) = \frac{E(v \to a)}{E(v)},\ \hat{e}_v(a,b) = \frac{E(v \to ab)}{E(v)}. \quad (2)$$

Note that the expected count correctly corresponding to its nonterminal type must be substituted for the above equations. In summary, the inside-outside algorithm is as follows:

**Algorithm 2 (**Inside-Outside**).**

**Initialization:** Pick arbitrary probability parameters of the model.

**Iteration:** Calculate the new probability parameters using (2). Calculate the new log likelihood $\sum_{r=1}^{N} \log P(w^{(r)} \mid \theta)$ of the model.

**Termination:** Stop if the change in log likelihood is less than predefined threshold. □

## 4. Experimental Results

### 4.1 Data for Experiments

The data sets for experiments were taken from an RNA family database called "Rfam" (version 7.0) [6] which is a database of multiple sequence alignment and covariance models [5] representing non-coding RNA families. We selected three viral RNA families with pseudoknot annotations named Corona_pk3 (Corona), HDV_ribozyme (HDV) and Tombus_3_IV (Tombus) (see **Table 2**). Corona_pk3 has a simple pseudoknotted structure, whereas HDV_ribozyme and Tombus_3_IV have more complicated structures with pseudoknot.

### 4.2 Implementation

We specified a particular SMCFG $G_R$ by utilizing secondary structure annotation of each family. Rules were determined by consider-

**Table 2** Three RNA families from Rfam ver. 7.0.

| Family | Range of length | # of annotated sequences | # of test sequences |
|---|---|---|---|
| Corona_pk3 | 62–64 | 14 | 10 |
| HDV_ribozyme | 87–91 | 15 | 10 |
| Tombus_3_IV | 89–92 | 18 | 12 |

**Table 3** Prediction results.

| Family | Precision [%] | | | Recall [%] | | | CPU time [sec] | | |
|---|---|---|---|---|---|---|---|---|---|
| | Average | Min | Max | Average | Min | Max | Average | Min | Max |
| Corona_pk3 | 99.4 | 94.4 | 100.0 | 99.4 | 94.4 | 100.0 | 27.8 | 26.0 | 30.4 |
| HDV_ribozyme | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 252.1 | 219.0 | 278.4 |
| Tombus_3_IV | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 244.8 | 215.2 | 257.5 |

Corona_pk3 (EMBL accession #: X51325)

[Trusted structure in Rfam]

```
CUAGUCUUAUACACAAUGGUAAGCCAGUGGUAGUAAAGGUAUAAGAAAUUUGCUACUAUGUUA
    [[[[[[[[                (((  (((((((  ]]]]]]]]   )))))))  )))
```

[Prediction by SMCFG]

```
CUAGUCUUAUACACAAUGGUAAGCCAGUGGUAGUAAAGGUAUAAGAAAUUUGCUACUAUGUUA
    [[[[[[[[                ((((((((((   ]]]]]]]]    ((((((((((
```

**Fig. 5** Comparison of a prediction result with a trusted structure in Rfam.

**Table 4** Comparison between SMCFG and PSTAG.

| Model | Average precision [%] | | | Average recall [%] | | |
|---|---|---|---|---|---|---|
| | Corona | HDV | Tombus | Corona | HDV | Tombus |
| SMCFG | 99.4 | 100.0 | 100.0 | 99.4 | 100.0 | 100.0 |
| PSTAG | 95.5 | 95.6 | 97.4 | 94.6 | 94.1 | 97.4 |

ing consensus secondary structure. Probability parameters were estimated in a few selected sequences by the simplest pseudocounting method known as the Laplace's rule [4]: to add one extra count to the true counts for each base configuration observed in a few selected sequences. Note that the inside-outside algorithm was not used in the experiments. The other sequences in the alignment were used as the test sequences for prediction (see Table 2). We implemented the CYK algorithm with traceback in ANSI C on a machine with Intel Pentium D CPU 2.80 GHz and 2.00 GB RAM. Straightforward implementation gives rise to a serious problem of lack of memory space due to the higher order dynamic programming matrix (remember that the space complexity of the CYK algorithm is $O(mn^4)$). The dynamic programming matrix in our specified model is sparse, and therefore, we successfully implemented the matrix as a hash table storing only nonzero probability values (equivalently, finite values of the logarithm of probabilities).

### 4.3 Tests

We tested prediction accuracy by calculating precision and recall (sensitivity), which are the ratio of the number of correct base pairs predicted by the algorithm to the total number of predicted base pairs, and the ratio of the number of correct base pairs predicted by the algorithm to the total number of base pairs specified by the trusted annotation, respectively. The results are shown in **Table 3**. A nearly correct prediction (94.4% precision and recall) for Corona_pk3 is shown in **Fig. 5** where underlined base pairs agree with trusted ones. The secondary structures predicted by our algorithm agree very well with the trusted structures. The running time of prediction in Corona_pk3 is much shorter than that of prediction in HDV_ribozyme and Tombus_3_IV since every sequence in Corona_pk3 can be generated by rules without BIFURCATION nonterminals. In this case, the time complexity of the CYK algorithm is $O(m^2n^4)$.

### 4.4 Comparison with PSTAG

We compared the prediction accuracy of our SMCFG algorithm with that of PSTAG algorithm [10] (see **Table 4**). PSTAGs, as we have mentioned before, are proposed for modeling pairwise alignment of RNA sequences with pseudoknots and assign a probability to each

alignment of TAG derivation trees. PSTAG algorithm, based on dynamic programming, calculates the most likely alignment for the pair of TAG derivation trees where one of them is in the form of an unfolded sequence and the other is a TAG derivation tree for known structure. SMCFG method is at least comparable to PSTAG method in the same test sets.

## 5. Discussion

In the computational experiments using SMCFG, we obtained good prediction results in accuracy and we did not trained probability parameters using the inside-outside algorithm any more. Part of the reason for success of prediction without training is that we were able to obtain good structural alignment from the database. The word "good" means that every trusted structure is little different from consensus structure and the number of gaps in alignment is relatively few. In fact, an earlier experimental results, omitted in this paper, showed only 76.6 % average precision and recall in Corona_pk3 and 95.7 % in Tombus_3_IV. We should notice that there are more gaps in the alignment of Corona_pk3 than that of Tombus_3_IV. Changing rules in such a way that DELETE rules are not successively used after the terminating rule $W_v \rightarrow (\varepsilon, \varepsilon)$, we can obtain the present results shown in Table 3. Hence, prediction accuracy will depend on the way to construct rules. We think that the most sensitive factor for prediction accuracy will be the number of consecutive gaps in alignment.

PSTAG method aligns an unfolded sequence with a derivation tree representing trusted structure. In SMCFG, rules are constructed according to a consensus structure and then the most probable derivation tree is calculated. In this sense, SMCFG and PSTAG have a common property that both of them take structural alignment into consideration implicitly or explicitly. Time and space complexities of SMCFG algorithm have the same order as those of PSTAG algorithm, whereas SMCFG algorithm consumes less memory than PSTAG algorithm since the dynamic programming matrix of SMCFG algorithm is sparse. This greatly contributes to practicability in computational structure prediction.

It is not certain that the differences in precision and recall between SMCFG and PSTAG are statistically significant since the number of analyzed data sets is small. SMCFGs can have arbitrary number of nonterminals and rules. On the other hand, PSTAG method takes three finite states into account, representing match, insertion and deletion states. Here, we regard nonterminals as states and rule application as state transitions [4]. The difference of the number of finite states may affect prediction accuracy.

## 6. Conclusion

In this paper, we have proposed a probabilistic model named SMCFG, and designed a polynomial time parsing and a parameter estimation algorithm for the specific SMCFG. Moreover, we have demonstrated computational experiments of RNA secondary structure prediction with pseudoknots using SMCFG parsing algorithm, which show good performance in accuracy.

Comparing with other prediction methods such as a thermodynamical approach, stochastic grammars have an advantage in easily modeling RNA secondary structure we would like to analyze and training probability parameters. We should notice that there is a trade-off between prediction accuracy and cost for constructing an initial grammar.

**Acknowledgments** This work is supported in part by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science (JSPS). The first author thanks JSPS Research Fellowships for Young Scientists for their generous financial assistance. The authors thank Dr. Yoshiaki Takata for his useful comments on implementation of high dimensional dynamic programming.

## References

1) Akutsu, T.: Dynamic Programming Algorithms for RNA Secondary Structure Prediction with Pseudoknots, *Discrete Applied Mathematics*, Vol.104, pp.45–62 (2000).
2) Brown, M. and Wilson, C.: RNA Pseudoknot Modeling Using Intersections of Stochastic Context Free Grammars with Applications to Database Search, *Proc. Pacific Symposium on Biocomputing*, pp.109–125 (1996).
3) Cai, L., Malmberg, R.L. and Wu, Y.: Stochastic Modeling of RNA Pseudoknotted Structures: A Grammatical Approach, *Bioinformatics*, Vol.19, suppl.1, pp.i66–i73 (2003).
4) Durbin, R., Eddy, S.R., Krogh, A. and Mitchison, G.: *Biological Sequence Analysis*, Cambridge University Press (1998).
5) Eddy, S.R. and Durbin, R.: RNA Sequence

Analysis Using Covariance Models, *Nuc. Acids Res.*, Vol.22, No.11, pp.2079–2088 (1994).
6) Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A. and Eddy, S.R.: Rfam: An RNA Family Database, *Nuc.Acids Res.*, Vol.31, No.1, pp.439–441 (2003).
7) Kasami, T., Seki, H. and Fujii, M.: Generalized Context-Free Grammar and Multiple Context-Free Grammar, *IEICE Trans. Inf. & Syst.*, Vol.J71-D, No.5, pp.758–765 (1988). (in Japanese).
8) Kato, Y., Seki, H. and Kasami, T.: On the Generative Power of Grammars for RNA Secondary Structure, *IEICE Trans. Inf. & Syst.*, Vol.E88-D, No.1, pp.53–64 (2005).
9) Kato, Y. and Seki, H.: Stochastic Multiple Context-Free Grammar for RNA Pseudoknot Modeling, *NAIST Info. Sci. Tech. Rep.*, NAIST-IS-TR2006002 (2006).
10) Matsui, H., Sato, K. and Sakakibara, Y.: Pair Stochastic Tree Adjoining Grammars for Aligning and Predicting Pseudoknot RNA Structures, *Bioinformatics*, Vol.21, No.11, pp.2611–2617 (2005).
11) Rivas, E. and Eddy, S.R.: A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots, *J. Mol. Biol.*, Vol.285, pp.2053–2068 (1999).
12) Rivas, E. and Eddy, S.R.: The Language of RNA: A Formal Grammar that Includes Pseudoknots, *Bioinformatics*, Vol.16, No.4, pp.334–340 (2000).
13) Sakakibara, Y., Brown, M., Hughey, R., Mian, I.S., Sjölander, K., Underwood, R.C. and Haussler, D.: Stochastic Context-Free Grammars for tRNA Modeling, *Nuc. Acids Res.*, Vol.22, pp.5112–5120 (1994).
14) Seki, H., Matsumura, T., Fujii M. and Kasami, T.: On Multiple Context-Free Grammars, *Theor. Comput. Sci.*, Vol.88, pp.191–229 (1991).
15) Uemura, Y., Hasegawa, A., Kobayashi, S. and Yokomori, T.: Tree Adjoining Grammars for RNA Structure Prediction, *Theor. Comput. Sci.*, Vol.210, pp.277–303 (1999).

## Appendix

### A.1 Outside Algorithm

**Algorithm 3 (**Outside**).**
**Initialization:**
$\beta_1(1,n) \leftarrow 1$

**Iteration:**
**for** $i \leftarrow 1$ **to** $n+1$, $j \leftarrow n$ **downto** $i-1$, $k \leftarrow j+1$ **to** $n+1$, $l \leftarrow n$ **downto** $k-1$, $v \leftarrow 1$ **to** $m$

**do if** $\mathrm{type}(v) = \mathrm{S}$ **and** $W_y \to C_1[W_v, W_z]$
**then** $\beta_v(i,j)$
$$\leftarrow \sum_{h=j}^{n} \sum_{k'=h+1}^{n+1} \sum_{l'=k'-1}^{n} \beta_y(i,h,k',l') \alpha_z(j+1,h,k',l')$$
**if** $\mathrm{type}(v) = \mathrm{S}$ **and** $W_y \to C_2[W_v, W_z]$
**then** $\beta_v(i,j)$
$$\leftarrow \sum_{h=1}^{i} \sum_{k'=j+1}^{n+1} \sum_{l'=k'-1}^{n} \beta_y(h,j,k',l') \alpha_z(h,i-1,k',l')$$
**if** $\mathrm{type}(v) = \mathrm{S}$ **and** $W_y \to C_3[W_v, W_z]$
**then** $\beta_v(i,j)$
$$\leftarrow \sum_{h=1}^{i} \sum_{k'=h-1}^{i-1} \sum_{l'=j}^{n} \beta_y(h,k',i,l') \alpha_z(h,k',j+1,l')$$
**if** $\mathrm{type}(v) = \mathrm{S}$ **and** $W_y \to C_4[W_v, W_z]$
**then** $\beta_v(i,j)$
$$\leftarrow \sum_{h=1}^{i} \sum_{k'=h-1}^{i-1} \sum_{l'=k'+1}^{i} \beta_y(h,k',l',j) \alpha_z(h,k',l',i-1)$$
**if** $\mathrm{type}(v) \neq \mathrm{S}$ **and** $W_y \to C_1[W_z, W_v]$
**then** $\beta_v(i,j,k,l)$
$$\leftarrow \sum_{h=1}^{i} \beta_y(h,j,k,l) \alpha_z(h,i-1)$$
**if** $\mathrm{type}(v) \neq \mathrm{S}$ **and** $W_y \to C_2[W_z, W_v]$
**then** $\beta_v(i,j,k,l)$
$$\leftarrow \sum_{h=j}^{k-1} \beta_y(i,h,k,l) \alpha_z(j+1,h)$$
**if** $\mathrm{type}(v) \neq \mathrm{S}$ **and** $W_y \to C_3[W_z, W_v]$
**then** $\beta_v(i,j,k,l)$
$$\leftarrow \sum_{h=j+1}^{k} \beta_y(i,j,h,l) \alpha_z(h,k-1)$$
**if** $\mathrm{type}(v) \neq \mathrm{S}$ **and** $W_y \to C_4[W_z, W_v]$
**then** $\beta_v(i,j,k,l)$
$$\leftarrow \sum_{h=l}^{n} \beta_y(i,j,k,h) \alpha_z(l+1,h)$$
**else** $\beta_v(i,j,k,l)$
$$\leftarrow \sum_{y \in \mathcal{P}_v} \beta_y(i-\Delta_y^{1L}, j+\Delta_y^{1R}, k-\Delta_y^{2L}, l+\Delta_y^{2R}) e_y(a_{i-\Delta_y^{1L}}, a_{j+\Delta_y^{1R}}, a_{k-\Delta_y^{2L}}, a_{l+\Delta_y^{2R}}) t_y(v)$$ □

### A.2 Expected Counts in Inside-Outside Algorithm

$$E(v \to y) = \sum_{r=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{h=i-1}^{j} \frac{1}{P(w^{(r)} \mid \theta)}$$

$$\beta_v^{(r)}(i,j)t_v(y)\alpha_y^{(r)}(i,h,h+1,j)$$

for type$(v) = $ S, and

$$E(v \to y) = \sum_{r=1}^{N}\sum_{i=1}^{n+1}\sum_{j=i-1}^{n}\sum_{k=j+1}^{n+1}\sum_{l=k-1}^{n}\frac{1}{P(w^{(r)} \mid \theta)} \beta_v^{(r)}(i,j,k,l)e_v(a_i,a_j,a_k,a_l)t_v(y) \alpha_y^{(r)}(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})$$

otherwise.

$$E(v \to a) = \sum_{r=1}^{N}\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j+1}^{n+1}\sum_{l=k-1}^{n}\frac{1}{P(w^{(r)} \mid \theta)} \delta(a_i^{(r)} = a)\beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l)$$

for type$(v) = $ U$_{1\mathrm{L}}$,

$$E(v \to a) = \sum_{r=1}^{N}\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j+1}^{n+1}\sum_{l=k-1}^{n}\frac{1}{P(w^{(r)} \mid \theta)} \delta(a_j^{(r)} = a)\beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l)$$

for type$(v) = $ U$_{1\mathrm{R}}$,

$$E(v \to a) = \sum_{r=1}^{N}\sum_{i=1}^{n-1}\sum_{j=i-1}^{n-1}\sum_{k=j+1}^{n}\sum_{l=k}^{n}\frac{1}{P(w^{(r)} \mid \theta)} \delta(a_k^{(r)} = a)\beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l)$$

for type$(v) = $ U$_{2\mathrm{L}}$,

$$E(v \to a) = \sum_{r=1}^{N}\sum_{i=1}^{n-1}\sum_{j=i-1}^{n-1}\sum_{k=j+1}^{n}\sum_{l=k}^{n}\frac{1}{P(w^{(r)} \mid \theta)} \delta(a_l^{(r)} = a)\beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l)$$

for type$(v) = $ U$_{2\mathrm{R}}$, and

$$E(v \to ab) = \sum_{r=1}^{N}\sum_{i=1}^{n-1}\sum_{j=i}^{n-1}\sum_{k=j+1}^{n}\sum_{l=k}^{n}\frac{1}{P(w^{(r)} \mid \theta)} \delta(a_i^{(r)} = a, a_l^{(r)} = b)\beta_v^{(r)}(i,j,k,l) \alpha_v^{(r)}(i,j,k,l)$$

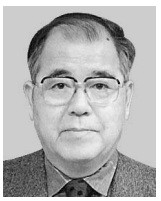
for type$(v) = $ P, where $\delta(C)$ is 1 if the condition $C$ in the parenthesis is ture, and 0 if $C$ is false.

(Received June 13, 2006)
(Accepted July 12, 2006)

(Communicated by *Tetsuo Shibuya*)

**Yuki Kato** received the M.E. degree in information processing from Nara Institute of Science and Technology in 2005. He is currently a doctoral course student of Graduate School of Information Science, Nara Institute of Science and Technology. Also, he is a Research Fellow of the Japan Society for the Promotion of Science. His current research interests include algorithms, formal language theory and their application to biological sequence analysis.

**Hiroyuki Seki** received the Ph.D. degree in information and computer sciences from Osaka University in 1987. He was with Osaka University as an Assistant Professor in 1990–1992 and an Associate Professor in 1992–1994. In 1994, he joined the faculty of Nara Institute of Science and Technology, where he has been a Professor since 1996. His current research interests include formal language theory and formal approach to software development.



**Tadao Kasami** received the Ph.D. degree in communication engineering from Osaka University in 1963. In 1963, he joined the faculty of Osaka University. He was a Professor at Osaka University in 1966–1994, at Nara Institute of Science and Technology in 1992–1998, and at Hiroshima City University in 1998–2003. He received the 1999 Claude E. Shannon Award from IEEE Information Theory Society. He is a life fellow of IEEE and an honorary member of IEICE.