Pacific Symposium on Biocomputing 7:163-174 (2002)

Pairwise RNA Structure Comparison with Stochastic Context-Free Grammars

I.Holmes and G.M.Rubin
Howard Hughes Medical Institute

Abstract

Pairwise stochastic context-free grammars ("Pair SCFGs") are powerful tools for finding conserved RNA structures, but unconstrained alignment to Pair SCFGs is prohibitively expensive. We develop versions of the Pair SCFG dynamic programming algorithms that can be conditioned on precomputed structures, significantly reducing the time complexity of alignment. We have implemented these algorithms for general Pair SCFGs in software that is freely available under the GNU Public License.

1 Introduction

Stochastic Context-Free Grammars (SCFGs) are powerful tools for RNA structure prediction and genefinding[1,2]. However, they are expensive to use. For two sequences of lengths L and M, simultaneous alignment and structure comparison using SCFGs has time complexity $\mathcal{O}(L^3M^3)$; this is cubic compared to $\mathcal{O}(LM)$, the complexity of aligning the same two sequences to a hidden Markov model[3,4]. Such computational power is beyond the reach of most labs.

One way to reduce the time complexity of these algorithms is to constrain the analysis, by supplying either the primary sequence alignment or the secondary structure assignment. Algorithms for the latter task (alignment of supplied structures) have been described[5,6] and implemented in the `Vienna` package[7]. These algorithms require manual specification of the scoring scheme. It is desirable to place such algorithms in a formal framework wherein the scoring scheme can be reliably optimised from a "training set" of trusted structural alignments.

We here present dynamic programming algorithms for two-sequence SCFGs that use devices called "fold envelopes" to restrict the set of base-pairings that the recursion is allowed to consider. These fold envelopes may be based on pre-computed structures. In extreme cases (when the parse trees are deep, e.g. if the structures contain long stem loops) the reduced running time can be as low as $\mathcal{O}(LM)$. One can also easily build "dummy" fold envelopes that reproduce the full, unconstrained algorithm. We describe an implementation of this algorithmic toolkit that works for any SCFG and is freely available under the terms of the GNU Public License[8].

2 Algorithms

SCFGs (sometimes called "Single SCFGs") are flexible models for RNA sequences allowing nested covariation[1]. Pair SCFGs are a generalisation of Single SCFGs, yielding joint probabilities for two sequences at once. In order to reduce the time-complexity of dynamic programming (DP) algorithms for Pair SCFGs, we will start with the analogous treatment for Single SCFGs.

The DP algorithms of interest include the Inside algorithm, the Cocke-Younger-Kasami (CYK) algorithm and the Inside-Outside algorithm[1]. The Inside algorithm calculates the likelihood of the sequence according to the SCFG, summed over all possible parses of the sequence; the CYK algorithm finds the maximum likelihood parse of the sequence; and the Inside-Outside algorithm finds the expected number of times that each grammar production is used, with the expectation taken over the posterior distribution of parses.

These algorithms (for Single SCFGs) work by calculating the partial likelihood of all substrings of the observed sequence, starting with zero-length substrings and working up to the full length. Since the algorithms compute likelihoods for all substrings, their memory usage is $\mathcal{O}(L^2)$ for a sequence of length L. The running time is higher at $\mathcal{O}(L^3)$, since an extra factor of L is incurred in combining adjacent substrings[1].

If the secondary structure of the RNA sequence is already known, a faster approach is to compute *conditional likelihoods* (for the Inside and CYK algorithms) or *conditional expectations* (for the Inside-Outside algorithm), where the given condition is the secondary structure. Rather than iterating over all the substrings, the conditional algorithms only iterate over substrings consistent with the given set of base-pairings.

The situation for Pair SCFGs is directly analogous. Rather than iterating over every *pair* of substrings of the two sequences, we can restrict the algorithms to a limited set consistent with precomputed secondary structures.

We generalise this idea by giving versions of the Inside, CYK and Inside-Outside algorithms for Pair SCFGs that are restricted to any valid sets of substrings of the two RNA sequences. We use the term "fold envelope" to describe such a set of substrings for any one sequence. The problem of calculating likelihoods conditioned on secondary structure is then reduced to one of computing the appropriate fold envelopes. We begin with some notation.

2.1 Notation: Pair SCFG

We deliberately follow the Covariance Model notation of Eddy and Durbin[1]. Although the use of Chomsky normal form would simplify the mathematics, this form is less appropriate for biological sequence analysis.

A Pair Stochastic Context-Free Grammar emits symbols in two sequences, X and Y. The terminals for these symbols are x_i and y_i respectively. For RNA, these take the values '`A`', '`C`', '`G`' and '`U`'. There is also a silent terminal, ϵ, that is only generated by nonterminals of type 'E' (see below).

The grammar has M different nonterminals denoted by $W_1, \ldots, W_M$. Let u and v be indices for states W_u and W_v. There are eighteen different types of state, with properties described in Table 1. These include E (End), N (Null), B (Bifurcation) and fifteen different types of emit state. The emit states each have two-letter identifiers of the form AB, where A denotes the emission in sequence X and B the emission in sequence Y. (For example: states of type 'PL' emit a left-right base pair in sequence X and a single leftwise base in sequence Y, whereas states of type 'NR' emit nothing in sequence X and a single rightwise base in sequence Y.) State W_1 is the "start" state and is always of type N. State W_M is the "end" state and is always of type E; in fact, W_M is the only state that can be of type E. We define s_u to be the *state type* of W_u, taking one of the eighteen values from the first column of Table 1;

The emission and transition probabilities for state u are given by $e_u(\cdot)$ and $t_u(\cdot)$ respectively. We define numbers Δ_u^{XL}, Δ_u^{XR}, Δ_u^{YL} and Δ_u^{YR} which are the number of symbols emitted to left and right in sequences X and Y.

We also define $\mathcal{C}_u$, the *children* of W_u (being the list of indices v for the states W_v that W_u can make a transition to) and $\mathcal{P}_u$, the *parents* of W_u (being the list of indices of states that make a transition to W_u). Bifurcating states (type B) always transit with probability 1 to two Null states, W_{l_u} (left) and W_{r_u} (right). It is possible for l_u to be the same as r_u. The child list $\mathcal{C}_u$ for a bifurcating state is defined to be (l_u, l_r). The parent lists $\mathcal{P}_{l_u}$ and $\mathcal{P}_{r_u}$ do *not* include u; instead, each Null state v has an associated left-parent list $\mathcal{P}_v^L = \{u : s_u = \mathrm{B}, l_u = v\}$ and a right-parent list $\mathcal{P}_v^R = \{u : s_u = \mathrm{B}, r_u = v\}$. This treatment of bifurcations differs slightly from the Covariance Model of Eddy *et al.*

We require that Pair SCFG's have no *null cycles*. That is to say, there is no sequence of productions that starts from an N or B state and returns to the same state without emitting any residues in either X or Y.

For convenience, we define two orderings on the state indices $1 \ldots M$. These are represented by the lists $\mathcal{F}_O$ and $\mathcal{F}_I$. The *outside fill order* $\mathcal{F}_O$ lists all the emitting states, followed by the non-emitting states sorted in topological order (i.e. parents before children). Conversely, the *inside fill order* $\mathcal{F}_I$ lists the emitting states followed by the non-emitting states sorted in *reverse* topological order (i.e. children before parents).

2.2 *Notation: Fold Envelope*

Let X be a sequence of length L, whose i'th base is x_i (where i starts at zero, i.e. $0 \le i < L$). Let $X_{i..j}$ be the subsequence of X running from base i to base $j-1$, so that $X_{0..L}$ is the full sequence and $X_{i..i}$ is an empty sequence. In general there are $L+1$ empty subsequences (from $X_{0..0}$ to $X_{L..L}$) and $\frac{1}{2}(L+1)(L+2)$ subsequences in total.

Suppose we have a parse tree of nonterminals aligning X to a SCFG as described in section 2.1 (we discount Y-emissions for the moment, pretending that the grammar is a Single SCFG). Then each node of the parse tree, together with its children, accounts for some subsequence $X_{i..j}$ of X (this subsequence is called the *inside* sequence). Likewise, the parents, siblings and cousins of this node account for the subsequences $X_{0..i}$ and $X_{j..L}$. (the *outside* sequence).

If the nonterminal at this node is W_u, where $s_u \neq \mathrm{B}$, then the subsequence for the immediate child node will be $X_{i+\Delta_u^{\mathrm{XL}}..j-\Delta_u^{\mathrm{XR}}}$. If $s_u = \mathrm{B}$ (i.e. W_u is a bifurcation) then the subsequences for the two child nodes will be $X_{i..k}$ and $X_{k..j}$ for some k, where $i \le k \le j$.

In computing the full dynamic programming matrix for a SCFG, one typically considers all subsequences $X_{i..j}$, all emissions $X_{i+\Delta_u^{\mathrm{XL}}..j-\Delta_u^{\mathrm{XR}}}$ and all bifurcations ($X_{i..k}$, $X_{k..j}$). Our intent is to consider a reduced set of subsequences, rather than the full $\frac{1}{2}(L+1)(L+2)$. In order to manage this, we formally enumerate these allowed subsequences and their associated emissions and bifurcations. The enumeration is called a *fold envelope*.

The fold envelope $\mathcal{E}$ consists of N ordered subsequences. The n'th subsequence is $X_{i_n..j_n}$, where n starts from zero (i.e. $0 \le n < N$). If a subsequence $X_{i..j}$ is in $\mathcal{E}$, then we define $n[X_{i..j}]$ to be its index within $\mathcal{E}$ (in other words, $n[X_{i_m..j_m}] \equiv m$). For all subsequences $X_{i..j}$, that are *not* in the envelope, $n[X_{i..j}]$ is defined to be the "illegal" value $\emptyset$ (which we typically represent as -1 in actual code).

The ordering is from inside to outside, so that

$$X_{i..j} \in \mathcal{E}, X_{k..l} \in \mathcal{E}, i \ge k, j \le l \Rightarrow n[X_{i..j}] \le n[X_{k..l}]$$

For each subsequence $X_{i_n..j_n}$ in the envelope, we precompute the following inward and outward *emission connections*:

$$c_{\text{in}}(n, \Delta^{\mathrm{L}}, \Delta^{\mathrm{R}}) = n\left[X_{i_n+\Delta^{\mathrm{L}},j_n-\Delta^{\mathrm{R}}}\right]$$

$$c_{\text{out}}(n, \Delta^{\mathrm{L}}, \Delta^{\mathrm{R}}) = n\left[X_{i_n-\Delta^{\mathrm{L}},j_n+\Delta^{\mathrm{R}}}\right]$$

where Δ^{L} and Δ^{R} can take values from $\{0, +1\}$. Some of these connections may equal $\emptyset$ if the target subsequence is not in the envelope.

We also pre-compute lists of valid inward, outward-left and outward-right *bifurcation connections*:

$$b_{\text{in}}(n) = \{(n_L, n_R) : i_{n_L} = i_n, j_{n_L} = i_{n_R}, j_{n_R} = j_n\}$$

$$b_{\text{outl}}(n) = \{(n_O, n_L) : i_{n_L} = i_{n_O}, j_{n_L} = i_n, j_n = j_{n_O}\}$$

$$b_{\text{outr}}(n) = \{(n_O, n_R) : i_n = i_{n_O}, j_n = i_{n_R}, j_{n_R} = j_{n_O}\}$$

Each element in a bifurcation connection list is a pair of subsequence indices. Together with subsequence n, these subsequences form a bifurcation triplet, i.e. *(outside,inside-left,inside-right)*. In contrast to the emission connections, the subsequence indices in the bifurcation connection lists are guaranteed not to be $\emptyset$.

An envelope is called *global* if it contains (i) all empty subsequences, (ii) the full-length sequence and (iii) at least one parse tree connecting the full-length sequence to one or more empty subsequences via emission or bifurcation connections. Note that by the inside-outside ordering condition, the full-length sequence has to be the last subsequence in a global envelope.

There exists an algorithm to calculate the appropriate fold envelope for a given structure in time that is linear in the length of the fold envelope (Holmes and Rubin, unpublished).

For the pairwise dynamic programming algorithms described below, we need two global envelopes, one for each sequence. We differentiate between these two envelopes by using apostrophes for envelope Y, i.e. the appropriate envelope variables for sequence X are $\{\mathcal{E}, N, L, i, j, n, c, b\}$ and for sequence Y $\{\mathcal{E}', N', L', i', j', n', c', b'\}$.

2.3 The Conditional Inside algorithm

The Conditional Inside algorithm, shown in Figure 1, recursively calculates $\alpha_u(n, n')$, the total likelihood for all joint parses rooted at nonterminal W_u of the two inside subsequences n and n'. The full likelihood of the two sequences is $\alpha_1(N, N')$.

For convenience, we define $\alpha_u(n, \emptyset) = \alpha_u(\emptyset, n') = 0$ for all u, n, n'. Also for convenience, we follow Durbin *et al* in using the notation $e_u(x_i, x_j, y_k, y_l)$ for all emission probabilities, even for nonterminals W_u that emit fewer than four symbols. Thus, for states of type N, $e_u(x_i, x_j, y_k, y_l) \equiv 1$; for states of type LN, $e_u(x_i, x_j, y_k, y_l) \equiv e_u(x_i)$; for states of type PN, $e_u(x_i, x_j, y_k, y_l) \equiv e_u(x_i, x_j)$; for states of type NP, $e_u(x_i, x_j, y_k, y_l) \equiv e_u(y_k, y_l)$ and so on (as per Table 1).

2.4 The Conditional Outside algorithm

To estimate counts for production usage conditioned on RNA structure, we need the Conditional Inside-Outside algorithm. The first half of this algorithm (the Conditional Inside algorithm) was already described in section 2.3. We now describe the second half.

The Conditional Outside algorithm shown in Figure 2, recursively calculates $\beta_u(n, n')$, the total likelihood for all joint parses rooted at nonterminal W_u of the outside subsequences n and n'. (Recall that the outside subsequence n of sequence X consists of the two subsequences $X_{0..i_n}$ and $X_{j_n..L}$. Likewise, the outside subsequence n' of sequence Y consists of the two subsequences $Y_{0..i'_{n'}}$ and $Y_{j'_{n'}..L'}$.)

As in section 2.3, we define $\beta_u(n, \emptyset) = \beta_u(\emptyset, n') = 0$ for all u, n, n'. We also use the notation $e_u(x_i, x_j, y_k, y_l)$ for all emission probabilities, as in that section.

We return to the Conditional Outside algorithm in section 2.7.

2.5 The Conditional CYK algorithm

The Conditional CYK algorithm, shown in Figure 3, recursively calculates $\gamma_u(n, n')$, the maximum likelihood for a joint parse rooted at nonterminal W_u of the inside subsequences n and n'.

As in section 2.3, we define $\gamma_u(n, \emptyset) = \gamma_u(\emptyset, n') = 0$ for all u, n, n'. We also use the notation $e_u(x_i, x_j, y_k, y_l)$ for all emission probabilities, as in that section.

On its own, the Conditional CYK algorithm may be used for database searching (i.e. to flag homologous structures). Together with the Conditional CYK Traceback algorithm (section 2.6) it can be used to find maximum-likelihood alignments of RNA structures.

2.6 The Conditional CYK Traceback algorithm

To recover the conditional maximum likelihood alignment of two RNA sequences following the Conditional CYK algorithm, we need to perform a Conditional CYK Traceback. The algorithm to do this is shown in Figure 4.

This algorithm steps through the $\gamma_u(n, n')$ likelihoods that were calculated by the Conditional CYK algorithm. Whereas the Conditional CYK algorithm goes in the *inside→outside* direction, Conditional CYK Traceback goes *outside→inside*, outputting the maximum likelihood alignment as it goes.

2.7 Estimating conditional counts

To estimate the conditional counts for emission and transition usage following the Conditional Inside and Outside algorithms, we use the following equations:

$$\begin{aligned}\hat{e}_u(x_i, x_j, y_k, y_l) &= \sum_{n \in \mathcal{E}} \sum_{n' \in \mathcal{E}'} \frac{\alpha_u(n,n')\beta_u(n,n')}{e_u(x_i,x_j,y_k,y_l)} \\ \hat{t}_u(v) &= \sum_{n \in \mathcal{E}} \sum_{n' \in \mathcal{E}'} \alpha_v(n,n')\beta_u(n,n')\end{aligned}$$

(The first of these equations strictly is valid only if $e_u(x_i, x_j, y_k, y_l) \neq 0$. In the special case that the emission probability is zero, the estimated count is also zero and this equation does not apply.)

The Conditional version of the expectation-maximization algorithm uses these counts, possibly together with prior distributions such as Dirichlet mixtures, to update the probability parameters for the Pair SCFG. This is repeated until the probability parameters (and the likelihood) do not improve any further.

2.8 Extension to Other Grammars

Dynamic programming recursions for higher-order grammars, such as the pseudoknot-capable grammar described by Rivas and Eddy[9], also involve iterations over subsets of the full sequence. (In the case of Rivas and Eddy's pseudoknot grammar, each subset of the full sequence is a pair of substrings with a "hole" between them.)

It would not present any theoretical difficulty to extend fold envelopes to such higher grammars. The central idea of fold envelopes is to encapsulate the iteration over alignable subsequences. This is just as feasible when the subsequences consist of substring pairs (as with the pseudoknot grammar) as when they consist of single substrings (as with standard Pair SCFG's).

Fold envelopes can also be used to obtain low-complexity recursions for lower-order grammars as special cases of higher-order recursions. For example, HMMs are (formally) a subset of SCFGs, yet one would usually not wish to re-use SCFG algorithms for HMMs, since the HMM algorithms only consider the $\sim L$ substrings of the form $X_{0..j}$, whereas the SCFG algorithms consider all $\sim L^3$ adjacent substring-pairs $X_{i..k}, X_{k..j}$. However, with a fold envelope, the SCFG recursion can be restricted to the substrings used by the HMM recursion.

3 Implementation

We have implemented the algorithms described here in `DART`, a freely available C++/Unix toolkit available from `www.biowiki.org/dart`.

`DART` includes implementations of the Conditional Inside, Conditional Outside and Conditional CYK/Traceback algorithms on Pair SCFGs of any topology with states as listed in Table 1. Fold envelopes may be calculated for any structure as well as for the full, unconstrained dynamic programming.

Since Single SCFG's are a subset of Pair SCFG's, `DART` can also be used for covariance modeling as described by Eddy and Durbin[10]. It can also emulate Single or Pair HMMs.

4 Discussion

In this paper, we have developed algorithms for conditional pairwise dynamic programming to stochastic context-free grammars within the bounds of *fold envelopes* based on precomputed structures. Fold envelopes are efficient and flexible: efficient because they limit the grammar's time complexity, flexible because conditioning on (say) a range of structures (rather than a single structure) merely involves computing an appropriate fold envelope and does not require redesigning the algorithms in Figures 1-4.

With the present investment in large-scale sequencing, computer-assisted identification of conserved RNA structure offers a genomics approach to studies of noncoding RNA genes[2], transcript localisation[11], poly-A stability[12], RNA-RNA duplexes[13] and many other areas. Perhaps the most attractive model from a theoretical viewpoint would be one that described the time-evolution of RNA structure including the effects of natural selection[3]. Such a model may be just around the corner: probabilistic evolutionary models for DNA and protein sequence analysis have made considerable progress in recent years[14,1,15]. In the meantime, we hope the present work may be useful to researchers interested in the evolutionary implications of RNA structure conservation in biology.

4.1 Acknowledgements

We would like to thank Sean Eddy, Elena Rivas, Erwin Frise, Eric Lai and Eliza McKenna for their useful input. This work was supported by the Howard Hughes Medical Institute.

1. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, UK, 1998.
2. S. R. Eddy. Noncoding RNA genes. *Current Opinion in Genetics and Development*, 9(6):695–699, 1999.
3. D. Sankoff and R. J. Cedergren. Simultaneous comparison of three or more sequences related by a tree. In D. Sankoff and J. B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*, chapter 9, pages 253–264. Addison-Wesley, Reading, MA, 1983.
4. V. Bafna, S. Muthukrishnan, and R. Ravi. Similarity between RNA strings. Technical report, Center for Discrete Mathematics and Theoretical Computer Science, 1996.
5. B. A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *Computer Applications in the Biosciences*, 4(3):387–393, 1988.
6. B. A. Shapiro and K. Z. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Computer Applications in the Biosciences*, 6(4):309–318, 1990.
7. S. Wuchty, W. Fontana, I. L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49(2):145–165, 1999.
8. The GNU Public License, 2000. Available in full from http://www.fsf.org/copyleft/gpl.html.
9. E. Rivas and S. R. Eddy. The language of RNA: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–340, 2000.
10. S. R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22:2079–2088, 1994.
11. A. Bashirullah, R. L. Cooperstock, and H. D. Lipshitz. RNA localization in development. *Annual Review of Biochemistry*, 67:335–394, 1998.
12. N. Proudfoot. Poly(A) signals. *Cell*, 64(4):671–674, 1991.
13. E. C. Lai and J. W. Posakony. Regulation of Drosophila neurogenesis by RNA:RNA duplexes? *Cell*, 93(7):1103–1104, 1998.
14. J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34:3–16, 1992.
15. I. Holmes and W. J. Bruno. Evolutionary HMMs: a Bayesian approach to multiple alignment. To appear in Bioinformatics, 2001., 2001.

State type (s_u)	Allowed productions	Δ_u^{XL}	Δ_u^{XR}	Δ_u^{YL}	Δ_u^{YR}	Emission probability	Transition probability
N	$W_u \to W_v$	0	0	0	0	1	$t_u(v)$
LN	$W_u \to x_i W_v$	1	0	0	0	$e_u(x_i)$	$t_u(v)$
RN	$W_u \to W_v x_j$	0	1	0	0	$e_u(x_j)$	$t_u(v)$
PN	$W_u \to x_i W_v x_j$	1	1	0	0	$e_u(x_i, x_j)$	$t_u(v)$
NL	$W_u \to y_k W_v$	0	0	1	0	$e_u(y_k)$	$t_u(v)$
LL	$W_u \to x_i y_k W_v$	1	0	1	0	$e_u(x_i, y_k)$	$t_u(v)$
RL	$W_u \to y_k W_v x_j$	0	1	1	0	$e_u(x_j, y_k)$	$t_u(v)$
PL	$W_u \to x_i y_k W_v x_j$	1	1	1	0	$e_u(x_i, x_j, y_k)$	$t_u(v)$
NR	$W_u \to W_v y_l$	0	0	0	1	$e_u(y_l)$	$t_u(v)$
LR	$W_u \to x_i W_v y_l$	1	0	0	1	$e_u(x_i, y_l)$	$t_u(v)$
RR	$W_u \to W_v x_j y_l$	0	1	0	1	$e_u(x_j, y_l)$	$t_u(v)$
PR	$W_u \to x_i W_v x_j y_l$	1	1	0	1	$e_u(x_i, x_j, y_l)$	$t_u(v)$
NP	$W_u \to y_k W_v y_l$	0	0	1	1	$e_u(y_k, y_l)$	$t_u(v)$
LP	$W_u \to x_i y_k W_v y_l$	1	0	1	1	$e_u(x_i, y_k, y_l)$	$t_u(v)$
RP	$W_u \to y_k W_v x_j y_l$	0	1	1	1	$e_u(x_j, y_k, y_l)$	$t_u(v)$
PP	$W_u \to x_i y_k W_v x_j y_l$	1	1	1	1	$e_u(x_i, x_j, y_k, y_l)$	$t_u(v)$
E	$W_u \to \epsilon$	0	0	0	0	1	1
B	$W_u \to W_{l_u} W_{r_u}$	0	0	0	0	1	1

Table 1: The eighteen types of nonterminal in a Pair Stochastic Context-Free Grammar and their associated production rules.

For $n = 0$ to $N-1$, $n' = 0$ to $N'-1$, $u \in \mathcal{F}_I$:

$$\alpha_u(n, n') = \begin{cases} s_u = \mathrm{E}: \quad \delta(i_n - j_n)\ \delta(i'_{n'} - j'_{n'}) \quad \text{where } \delta(x) = \begin{cases} 1 & \text{if} \quad x = 0 \\ 0 & \text{if} \quad x \neq 0 \end{cases} \\[2ex] s_u = \mathrm{B}: \quad \displaystyle\sum_{\substack{(n_L, n_R) \\ \in\ b_{\mathrm{in}}}} \sum_{\substack{(n'_L, n'_R) \\ \in\ b'_{\mathrm{in}}}} \alpha_{l_u}(n_L, n'_L)\ \alpha_{r_u}(n_R, n'_R) \\[2ex] \text{otherwise:} \\ \quad e_u(x_{i_n}, x_{j_n - 1}, y_{i'_{n'}}, y_{j'_{n'} - 1}) \displaystyle\sum_{v \in \mathcal{C}_u} t_u(v)\ \alpha_v\left(c_{\mathrm{in}}(n, \Delta_u^{\mathrm{XL}}, \Delta_u^{\mathrm{XR}}),\ c'_{\mathrm{in}}(n', \Delta_u^{\mathrm{YL}}, \Delta_u^{\mathrm{YR}})\right) \end{cases}$$

Figure 1: The Conditional Inside algorithm for Pair SCFGs.

For $n = N-1$ to 0, $n' = N'-1$ to 0, $u \in \mathcal{F}_O$:

$$\begin{aligned}\beta_u(n,n') = \quad & e_u(x_{i_n-1}, x_{j_n}, y_{i'_{n'}-1}, y_{j'_{n'}}) \left(\sum_{v \in \mathcal{P}_u} t_v(u)\ \beta_v \left(c_{\text{out}}(n, \Delta_u^{\text{XL}}, \Delta_u^{\text{XR}}),\ \ c'_{\text{out}}(n', \Delta_u^{\text{YL}}, \Delta_u^{\text{YR}})\right)\right. \\ & + \sum_{\substack{v \in \mathcal{L}_u, \\ s_v = B}} \sum_{\substack{(n_O, n_L) \\ \in\ b_{\text{outl}}}} \sum_{\substack{(n'_O, n'_L) \\ \in\ b'_{\text{outl}}}} \beta_v(n_O, n'_O)\ \alpha_{l_v}(n_L, n'_L) \\ & + \sum_{\substack{v \in \mathcal{R}_u, \\ s_v = B}} \sum_{\substack{(n_O, n_R) \\ \in\ b_{\text{outr}}}} \sum_{\substack{(n'_O, n'_R) \\ \in\ b'_{\text{outr}}}} \beta_v(n_O, n'_O)\ \alpha_{r_v}(n_R, n'_R)\end{aligned}$$

Figure 2: The Conditional Outside algorithm for Pair SCFGs.

For $n = 0$ to $N-1$, $n' = 0$ to $N'-1$, $u \in \mathcal{F}_I$:

$$\gamma_u(n,n') = \begin{cases} s_u = \text{E}: \quad \delta(i_n - j_n)\ \delta(i'_{n'} - j'_{n'}) \qquad \text{where } \delta(x) = \begin{cases} 1 & \text{if} \quad x = 0 \\ 0 & \text{if} \quad x \neq 0 \end{cases} \\ \\ s_u = \text{B}: \quad \max\limits_{\substack{(n_L, n_R) \\ \in\ b_{\text{in}}}} \max\limits_{\substack{(n'_L, n'_R) \\ \in\ b'_{\text{in}}}} \gamma_{l_u}(n_L, n'_L)\ \gamma_{r_u}(n_R, n'_R) \\ \text{otherwise:} \\ \quad e_u(x_{i_n}, x_{j_n-1}, y_{i'_{n'}}, y_{j'_{n'}-1}) \max\limits_{v \in \mathcal{C}_u} t_u(v)\ \gamma_v\left(c_{\text{in}}(n, \Delta_u^{\text{XL}}, \Delta_u^{\text{XR}}),\ \ c'_{\text{in}}(n', \Delta_u^{\text{YL}}, \Delta_u^{\text{YR}})\right) \end{cases}$$

Figure 3: The Conditional CYK algorithm for Pair SCFGs.

```
• Initialisation:
    – Let $u = 1$, $n = N - 1$, $n' = N' - 1$;    /* set traceback co-ordinates to
      start state */
    – Clear co-ordinates stack;
• Main loop:
    – Output co-ordinates $(u, n, n')$;
    – If $s_u$ = E    /* end state? */
        ∗ If co-ordinates stack is empty then exit;
        ∗ Pop co-ordinates $(u, n, n')$;
        ∗ Goto Main loop;
    – else if $s_u$ = B    /* bifurcation state? */
        ∗ Select $(n_L, n_R)$ from $b_{\texttt{in}}$ and $(n'_L, n'_R)$ from $b'_{\texttt{in}}$
          such that $\gamma_u(n, n') = \gamma_{l_u}(n_L, n'_L)\gamma_{r_u}(n_R, n'_R)$;
        ∗ Push co-ordinates $(r_u, n_R, n'_R)$;
        ∗ Set $(u, n, n')$ equal to $(l_u, n_L, n'_L)$;
        ∗ Goto Main loop;
    – else    /* emit or null state */
        ∗ Let $(m, m') = (c_{\text{in}}(n, \Delta_u^{\text{XL}}, \Delta_u^{\text{XR}}),\ c'_{\text{in}}(n', \Delta_u^{\text{YL}}, \Delta_u^{\text{YR}}))$;
        ∗ Select $v$ from $\mathcal{C}_u$ such that $\gamma_u(n, n') = t_u(v)\gamma_v(m, m')$;
        ∗ Set $(u, n, n')$ equal to $(v, m, m')$;
        ∗ Goto Main loop;
```

Figure 4: The Conditional CYK Traceback algorithm for Pair SCFGs.