NAIST-IS-DD0561011

Doctoral Dissertation

Formal Grammars for Describing RNA Pseudoknotted Structure and Their Application to Structure Analysis

Yuki Kato

February 1, 2007

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Yuki Kato

Thesis Committee:

Professor Hiroyuki Seki	(Supervisor)
Professor Yuji Matsumoto	(Co-supervisor)
Professor Shigehiko Kanaya	(Co-supervisor)
Associate Professor Yuichi Kaji	(Co-supervisor)

Formal Grammars for Describing RNA Pseudoknotted Structure and Their Application to Structure Analysis*

Yuki Kato

Abstract

Recently, much attention has been paid to the structure analysis of biologically important molecules such as nucleic acids and proteins. These structures are hierarchically classified into primary structure, secondary structure and tertiary structure. In this thesis, we focus on RNA (ribonucleic acid) secondary structure determined by interactions between mostly Watson-Crick complementary base pairs. Since base pairs in typical RNAs occur in a nested way, RNA secondary structures have been successfully modeled by context-free grammars (CFGs), and secondary structure prediction has been translated into a parsing problem. On the other hand, there are substructures called pseudoknots where some base pairs occur in a crossed fashion, which cannot be represented by CFGs. Therefore, several formal grammars have been proposed for describing RNA secondary structure including pseudoknots, such as simple linear tree adjoining grammars (SLTAGs), extended SLTAGs (ESLTAGs) and RNA pseudoknot grammars (RPGs). However, the relation between the generative power of each of these grammars has not been clarified so far.

The first aim of this thesis is to compare the generative power of the grammars mentioned above by identifying them as subclasses of multiple context-free grammars (MCFGs), which are natural extension of CFGs. More specifically, the following properties are shown: (1) the class of languages generated by RPGs agrees with the class of languages generated by MCFGs with dimension one or two and rank one or two; (2) the class of languages generated by ESLTAGs ($\mathcal{ESLTAL}$) coincides with the class of languages generated by MCFGs with degree five or less; (3) $\mathcal{ESLTAL}$ properly

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0561011, February 1, 2007.

includes the union of the class of languages generated by SLTAGs ($\mathcal{SLTAL}$) and the class of languages generated by CFGs; (4) $\mathcal{SLTAL}$ is a full trio; and (5) $\mathcal{ESLTAL}$ is a substitution closed full AFL. Considering these results, the class of ESLTAGs can be a candidate for the minimum grammars that can represent pseudoknots.

The latter part of this thesis is dedicated to analyzing RNA secondary structure with pseudoknots by using the subclass of MCFGs corresponding to the class of ESLTAGs. When we interpret structure prediction as parsing of the grammar, we face the problem that there may be many different derivation trees for an input RNA sequence. Therefore, we take a practical approach, where we extend a grammar to a probabilistic model and find the most likely derivation tree. In this thesis, the above subclass of MCFGs is extended to a probabilistic model called stochastic MCFGs (SMCFGs). We present a polynomial time parsing algorithm for finding the most probable derivation tree and a probability parameter estimation algorithm based on the EM algorithm. Several experimental results are shown where RNA pseudoknotted structure predictions were carried out for viral RNA families using the SMCFG parsing algorithm. Furthermore, we perform RNA gene finding for several genome sequences known to have RNA genes with pseudoknots. This is achieved by using the scanning algorithm based on the SMCFG parsing algorithm. These experimental results show very close to 100% accuracy.

Keywords:

multiple context-free grammar, tree adjoining grammar, RNA pseudoknot grammar, generative power, closure property, RNA secondary structure, pseudoknot

RNA シュードノット構造記述向き形式文法とその構造解析への応用*

加藤 有己

内容梗概

近年，核酸やタンパク質などの生物学的に重要な分子の構造解析が注目を浴びている．これらの構造は階層的に１次構造，２次構造，３次構造と分類される．本論文では，多くが Watson-Crick 相補塩基対間の相互作用で決定される，RNA (リボ核酸) の２次構造に焦点を当てる．典型的な RNA では塩基対が互いに入れ子になって現れるため，RNA ２次構造を文脈自由文法 (CFG) でモデル化し，２次構造予測を文法の構文解析に置き換える試みが行われてきた．一方で，いくつかの塩基対が交差して現れる，シュードノットと呼ばれる部分構造が存在し，CFG では表現できないことが知られている．そのため，線形接木文法 (SLTAG)，拡張 SLTAG (ESLTAG)，RNA シュードノット文法 (RPG) などの，シュードノットを含む RNA の２次構造を記述する形式文法がいくつか提案された．しかしながら，現在までこれらの文法の生成能力間の関係は明らかではなかった．

本論文の最初の目的は，上記文法の生成能力を比較することである．そのために，上記文法クラスを CFG の自然な拡張である多重文脈自由文法 (MCFG) の部分クラスとして同定した．具体的には以下を示した: (1) RPG が生成する言語のクラスは，次元が 2 以下，ランクが 2 以下の MCFG が生成する言語のクラスに一致する，(2) ESLTAG が生成する言語のクラス ($\mathcal{ESLTAL}$) は，自由度が 5 以下の MCFG が生成する言語のクラスに一致する，(3) $\mathcal{ESLTAL}$ は，SLTAG が生成する言語のクラス ($\mathcal{SLTAL}$) と CFG が生成する言語のクラスとの和集合を真に含む，(4) $\mathcal{SLTAL}$ は full trio である，(5) $\mathcal{ESLTAL}$ は代入のもとで閉じた full AFL である．これらの結果を考慮すれば，現在知られている形式文法の中でシュードノットを表現できる生成能力最小の文法は ESLTAG であると考えられる．

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DD0561011, 2007 年 2 月 1 日.

本論文の後半では，ESLTAG に対応する MCFG の部分クラスを用いて，シュードノットを含む RNA ２次構造を解析する手法について論じる．構造予測を文法の構文解析とみなすとき，入力 RNA 配列に対して一般に多くの導出木が存在するという問題に直面する．従って，実用的には文法を確率モデルに拡張し，確率最大の導出木を求めるアプローチをとる．本論文では，上記 MCFG の部分クラスを確率 MCFG (SMCFG) と呼ばれる確率モデルに拡張する．次に，多項式時間で確率最大の導出木を求める構文解析アルゴリズム及び EM アルゴリズムに基づく確率パラメータ推定アルゴリズムを与える．また，SMCFG の構文解析アルゴリズムを用いて，ウイルス性 RNA に対して２次構造予測を行った結果を示す．さらに，上記構文解析アルゴリズムに基づく走査アルゴリズムを用いて，シュードノットを持つ RNA 遺伝子が含まれているいくつかのゲノム配列に対して RNA 遺伝子発見を行った．これらの実験結果は 100%に近い予測精度を示している．

キーワード

多重文脈自由文法，接木文法，RNA シュードノット文法，生成能力，閉包性，RNA ２次構造，シュードノット

List of Publications

Journal Papers

(1) Y. Kato, H. Seki and T. Kasami, “On the generative power of grammars for RNA secondary structure,” IEICE Transactions on Information & Systems, vol. E88-D, no. 1, pp. 53–64, Jan. 2005.

(2) Y. Kato, H. Seki and T. Kasami, “RNA pseudoknotted structure prediction using stochastic multiple context-free grammar,” IPSJ Transactions on Bioinformatics, vol. 47, no. SIG17 (TBIO1), pp. 12–21, Nov. 2006.

International Conferences (Reviewed)

(3) Y. Kato, H. Seki and T. Kasami, “A comparative study on formal grammars for pseudoknots,” Proceedings of the 14th International Conference on Genome Informatics (GIW2003), pp. 470–471, Dec. 2003.

(4) Y. Kato, H. Seki and T. Kasami, “Subclasses of tree adjoining grammar for RNA secondary structure,” Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7), pp. 48–55, May 2004.

(5) Y. Kato, H. Seki and T. Kasami, “RNA structure prediction including pseudoknots based on stochastic multiple context-free grammar,” Proceedings of International Workshop on Probabilistic Modeling and Machine Learning in Structural and Systems Biology (PMSB2006), pp. 32–37, Jun. 2006.

(6) Y. Kato, H. Seki and T. Kasami, “Stochastic multiple context-free grammar for RNA pseudoknot modeling,” Proceedings of the 8th International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+8), pp. 57–64, Jul. 2006.

Other Workshops

(7) Y. Kato, H. Seki and T. Kasami, “On the generative power of grammars for RNA secondary structure,” IEICE Technical Report, COMP2003-75, Jan. 2004.

(8) Y. Kato and H. Seki, “Stochastic multiple context-free grammar for RNA structure analysis,” IPSJ SIG Technical Reports, 2005-BIO-3 (2), Dec. 2005.

(9) Y. Kato, “Stochastic multiple context-free grammar for RNA pseudoknot modeling,” Talk at International Workshop on Computational Approaches to Functional and Regulatory RNAs (RNA2006), Jul. 2006.

(10) Y. Kato, H. Seki and T. Kasami, “RNA pseudoknotted structure prediction using stochastic multiple context-free grammar,” IPSJ SIG Technical Reports, 2006-BIO-6 (5), Sep. 2006.

(11) Y. Kato, “Stochastic grammars and their application to biological sequence analysis,” Talk at the 17th Seminar on Algebra, Logic and Geometry in Informatics (ALGI17), Oct. 2006.

Acknowledgements

First and foremost, I wish to express my deepest thanks to Professor Hiroyuki Seki of Nara Institute of Science and Technology for his continuous support of the work. He advised me to study bioinformatics, suggested ideas and gave me beneficial advice, which greatly helped improve the quality of this study.

I am grateful to the members of the thesis committee: Professor Yuji Matsumoto, Professor Shigehiko Kanaya and Associate Professor Yuichi Kaji of Nara Institute of Science and Technology for their helpful suggestions and comments in reviewing the thesis.

I wish to express my gratitude to Professor Emeritus Tadao Kasami of Osaka University and Nara Institute of Science and Technology. He advised me to study RNA structure analysis using a grammar-based approach, and gave me helpful comments on an earlier draft of this thesis. Also, I often had invaluable time to discuss with him.

My special thanks are due to Professor Yoshihide Watanabe of Doshisha University. Since I was at Doshisha University, he has continuously encouraged me and given me helpful advice for continuing my studies.

I gratefully acknowledge fruitful discussions with Professor Sean R. Eddy and Professor Elena Rivas of Washington University, and Dr. Kengo Sato of National Institute of Advanced Industrial Science and Technology on several points in this thesis during my stay in Spain.

I wish to express my thanks to Dr. Yoshiaki Takata of Nara Institute of Science and Technology for his useful comments on programming.

I am also indebted to Professor Kiyoshi Asai of the University of Tokyo, Professor Yasubumi Sakakibara of Keio University and members of his laboratory, and Dr. Aki Hasegawa of RIKEN for their fruitful discussions.

Thanks are due to members of Laboratory for Foundations of Information Science of Nara Institute of Science and Technology, and members of Laboratory for Applied Mathematics of Doshisha University for their kindness and help.

I also wish to thank Research Fellowships of the Japan Society for the Promotion of Science for Young Scientists for their generous financial assistance.

Finally, I thank my family for supporting my research activities.

Contents

List of Figures

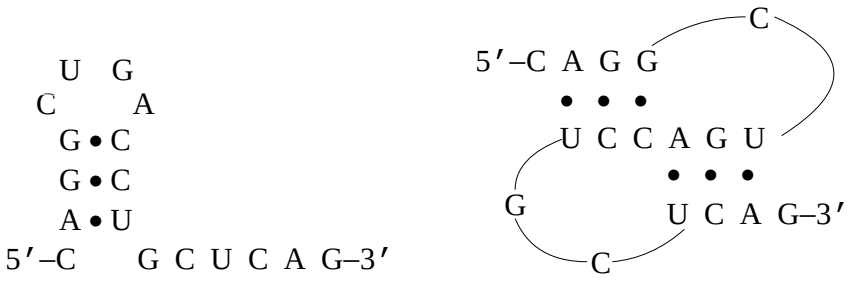
List of Tables

Chapter 1

Introduction

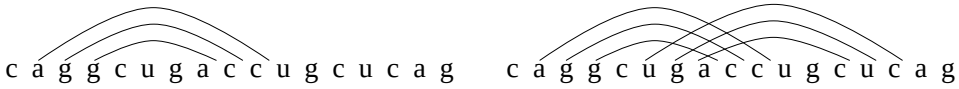
In recent years, genome sequences of various kinds of organisms have been experimentally determined, and many databases for biological sequences have been constructed. These sequences can be regarded as linear strings (primary structures) composed of four letters representing nucleotides for DNAs (deoxyribonucleic acids) and RNAs (ribonucleic acids), or twenty letters representing amino acids for proteins. The purpose of biological sequence analysis is to analyze genetic information from sequence data by using methods in informatics and statistics, including gene finding, homology searches and structure prediction. In particular, analyzing the structure of a biomolecule leads to the elucidation of its function since it is empirically known that if the structure of one molecule is similar to that of another, both functions will be similar. This motivates us to predict structures from biological sequences. This thesis concerns *RNA secondary structure* representing folding information.

RNA secondary structure is determined by interactions between mostly *Watson-Crick complementary base pairs* such as A-U and G-C. Figure 1.1 (a) shows a simple RNA secondary structure called a *stem loop*. If we connect the base pairs with the arcs on the RNA sequence, we can obtain another representation of the secondary structure shown in Figure 1.2 (a). Since base pairs in typical RNAs occur in a nested way like stem loops, RNA secondary structures have been successfully modeled by context-free grammars (CFGs), and secondary structure prediction has been translated into a parsing problem. Techniques based on the CYK (Cocke-Younger-Kasami) algorithm have been widely investigated [8, 9, 26]. On the other hand, there are substructures called *pseudoknots* where some base pairs occur in a crossed fashion (see Figures 1.1 (b) and



(a) Stem loop (b) Pseudoknot

Figure 1.1. Example of RNA secondary structure



(a) Stem loop (b) Pseudoknot

Figure 1.2. Arc depiction of Figure 1.1

1.2 (b)). They are found in several RNAs such as rRNAs, tmRNAs and viral RNAs. It has been recognized that pseudoknots play an important role in RNA functions such as ribosomal frameshifting and regulation of translation. However, CFGs cannot represent crossing dependency of base pairs in pseudoknotted structure due to the lack of generative power.

As for formal grammars whose generative power is stronger than that of CFGs, a significant concept was developed for syntax of natural language, which is called *mildly context-sensitive grammar* (mildly CSG) [13]. It has been widely recognized that the generative power of CFGs is not sufficient for syntax of natural language. For example, discontinuous structures such as respectively sentence construction cannot be represented by CFGs. For specifying the syntax of natural language, several grammars have been proposed, including *tree adjoining grammars* (TAGs) [11, 12], head grammars (HGs), linear indexed grammars (LIGs), combinatory categorical grammars (CCGs), linear context-free rewriting systems (LCFRSs) and *multiple context-free grammars* (MCFGs) [14, 15, 29]. The generative power of TAGs, HGs, LIGs and CCGs are known to be the same. Also, the generative power of LCFRSs and MCFGs are the same and strictly stronger than that of TAGs. The common features of these

grammars are as follows:

- The generative power of these grammars is strictly stronger than that of CFGs and strictly weaker than that of CSGs.
- Languages generated by these grammars can be recognized in polynomial time of the length of an input sequence. This contrasts with the fact that the recognition problem for CSGs is PSPACE-complete.
- These grammars inherit good mathematical properties of CFGs. For example, the classes of languages generated by these grammars are closed under union, concatenation, Kleene closure, homomorphism and intersection with regular languages.

In bioinformatics, several formal grammars have been proposed for fully describing RNA secondary structure including pseudoknots. In one pioneering paper [30], Uemura et al. defined two subclasses of TAGs called *simple linear TAGs* (SLTAGs) and *extended SLTAGs* (ESLTAGs), and argued that the class of ESLTAGs is appropriate for representing RNA pseudoknotted structure. Rivas and Eddy [24] provided keen observations on the representation of RNA secondary structure by a sequence with a single gap, and introduced a new class of grammars called *RNA pseudoknot grammars* (RPGs) for deriving sequences with gaps. These grammars have generative power stronger than CFGs, while recognition can be performed in polynomial time. However, the relation between the generative power of each of these grammars has not been clarified so far.

The first aim of this thesis is to compare the generative power of these grammars by identifying them as subclasses of MCFGs. This is partially motivated by an interest in formal language theory. As explained in detail below, it is interesting to compare the generative power of grammars for natural language syntax with that of grammars for biomolecule structure. An insight obtained in biological sequence analysis may result in significant progress in formal language theory. Another motivation comes from the observation that some techniques in formal language theory may contribute to a new analysis method in biology. If we can model a secondary structure of a particular type of biological sequence by a formal grammar for which a recognition method is well studied, we do not need to construct a prediction algorithm for that type of sequence

from scratch, and we can focus on how to provide appropriate scores or measures for grammar rules to obtain biologically realistic predictions.

The main contributions of the first part of this thesis are as follows:

(1) It is shown that the class of languages generated by RPGs agrees with the class of languages generated by MCFGs with dimension one or two and rank one or two, the class of languages generated by ESLTAGs ($\mathcal{ESLTAL}$) coincides with the class of languages generated by MCFGs with degree five or less, and $\mathcal{ESLTAL}$ properly includes the union of the class of languages generated by SLTAGs ($\mathcal{SLTAL}$) and the class of languages generated by CFGs.

(2) It is shown that $\mathcal{SLTAL}$ is a full trio and $\mathcal{ESLTAL}$ is a substitution closed full AFL.

The second part of this thesis is dedicated to analyzing RNA secondary structure with pseudoknots by using the subclass of MCFGs corresponding to the class of ESLTAGs. When we interpret structure prediction as parsing of the grammar, we face the problem that there may be many different derivation trees for an input RNA sequence. One practical approach to this problem is to extend a grammar to a probabilistic model and find the most likely derivation tree. Another is to take free energy minimization into account. The advantage of the latter approach is its prediction accuracy. However, it is not always easy to obtain sufficient thermodynamic information on sequences in experiments, and hence the former approach, based on a stochastic model, is often practical. Eddy and Durbin [9], and Sakakibara et al. [26] modeled RNA pseudoknot-free structure by using stochastic context-free grammars (stochastic CFGs or SCFGs). For pseudoknotted structure, Matsui et al. [20] proposed pair stochastic tree adjoining grammars (PSTAGs) based on ESLTAGs and tree automata for aligning and predicting pseudoknots, which showed good prediction accuracy. In this thesis, the above subclass of MCFGs is extended to a probabilistic model called *stochastic MCFGs* (SMCFGs). We present a polynomial time parsing algorithm for finding the most probable derivation tree and a probability parameter estimation algorithm based on the EM algorithm. Several experimental results are shown where RNA pseudoknotted structure predictions were carried out for three viral RNA families using the SMCFG parsing algorithm. These results show very close to 100% accuracy, and we can say that the SMCFG method is at least comparable to the PSTAG method in the same test sets. Fur-

thermore, by using the scanning algorithm based on the SMCFG parsing algorithm, we successfully perform RNA gene finding for several genome sequences known to have RNA genes with pseudoknots.

The remainder of this thesis is organized as follows. Chapter 2 surveys RNA secondary structure and multiple context-free grammar, which play a major role in this study. In Chapter 3, we clarify the relation between the generative power of grammars for RNA secondary structure including pseudoknots. Chapter 4 analyzes RNA pseudoknotted structure by using stochastic multiple context-free grammar. Chapter 5 concludes this thesis.

Chapter 2

Preliminaries

1. RNA Secondary Structure

As the central dogma of molecular biology claims, RNA is widely known as an intermediary messenger between DNA gene storing genetic information and protein determining its biological function. This is called mRNA and represented as an unstructured linear strand. Meanwhile, there exist many RNAs that are not translated into proteins such as rRNAs and tRNAs. They are called *non-coding RNAs* (ncRNAs) and fold into characteristic three-dimensional structures so that they have specific functions. In this thesis, we are concerned with ncRNAs and ncRNA is often written as RNA.

RNA is a high polymer consisting of four different nucleotides. They are abbreviated as A (adenine), C (cytosine), G (guanine) and U (uracil) [1]. A-U and G-C form hydrogen bonded base pairs, which is called *Watson-Crick complementary base pairs*. G-C pairs form three hydrogen bonds while A-U pairs form two hydrogen bonds, and therefore, the former is more stable than the latter. The other non-canonical base pairs also occur in RNA, where the most common pair is G-U that is thermodynamically favorable. The resulting base paired structure is called the *secondary structure*. Note that the secondary structure does not give information on the three-dimensional structure of a molecule. It is known that stacked base pairs twist to form double helices like DNA.

Each element of an RNA secondary structure is shown in Figure 2.1. A hydrogen bond is represented as a bullet. The continuous region of stacked base pairs is called

[1] In DNA, T (thymine) replaces U.

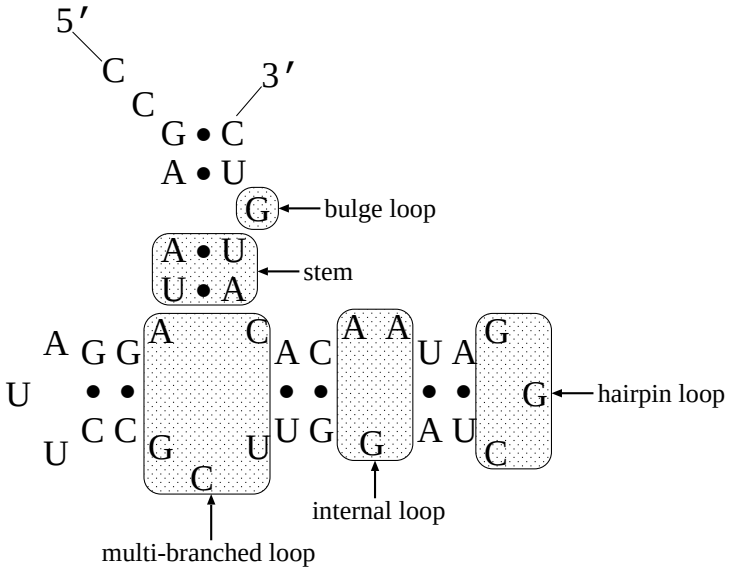


Figure 2.1. Fundamental elements of RNA secondary structure

a *stem*, and a single strand connecting a stem is called a *loop*. A simple substructure consisting of a single stem and a single loop is called a *stem loop* (see Figure 1.1 (a)). Loops are classified according to their positions to stems as follows. A loop at the edge of a stem is called a *hairpin loop*. A loop is called a *bulge loop* if a single strand occurs on one side of a stem, or *internal loop* if a single strand occurs on both sides of a stem. A loop is called a *multi-branched loop* if more than two stems radiate from the loop.

In typical RNA secondary structures, when we connect base pairs with arcs on the sequence, the arcs are hierarchically nested, that is, none of the arcs crosses each other (see Figure 1.2 (a)). Formally, let i and j be the positions of bases forming a base pair and i' and j' be the positions of bases forming another base pair. Base pairs are nested if and only if for all pairs (i, j) and (i', j'), either $i < i' < j' < j$ or $i' < i < j < j'$ holds. On the other hand, RNA substructures where arcs are crossing occurs, which is called *pseudoknots* (see Figures 1.1 (b) and 1.2 (b)). As in the definition of nested structure, a secondary structure is a pseudoknot if and only if there exist pairs (i, j) and (i', j') such that $i < i' < j < j'$ or $i' < i < j' < j$. Although the total number of base pairs forming pseudoknots is relatively smaller than the number of base pairs forming nested structures, we must take pseudoknot into consideration for three-dimensional structure prediction.

2. Multiple Context-Free Grammar

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

A *multiple context-free grammar* (MCFG) [14, 29] 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 as $\dim(A)$ 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 $(0 \leq i \leq k)$ 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 ith argument of f for $1 \leq i \leq k$. The hth component of the function value for $1 \leq h \leq d_0$, denoted by $f^{[h]}$, is defined as

$$f^{[h]}[\overline{x_1}, \ldots, \overline{x_k}] = \beta_{h0} z_{h1} \beta_{h1} z_{h2} \cdots z_{hv_h} \beta_{hv_h} \tag{2.1}$$

where $\beta_{hl} \in T^*$ $(0 \leq l \leq v_h)$ and $z_{hl} \in \{x_{ij} \mid 1 \leq i \leq k,\ 1 \leq j \leq d_i\}$ $(1 \leq l \leq v_h)$. The total number of occurrences of x_{ij} in the right hand sides of (2.1) from $h = 1$ through d_0 is at most one.

Each rule in P has the form of $A_0 \to f[A_1, \ldots, A_k]$ where $A_i \in N$ $(0 \leq i \leq k)$ and $f : (T^*)^{\dim(A_1)} \times \cdots \times (T^*)^{\dim(A_k)} \to (T^*)^{\dim(A_0)} \in F$. If $k \geq 1$, 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 $f^{[h]}[\,] = \beta_h$ $(1 \leq h \leq \dim(A_0))$ is simply written as $A_0 \to (\beta_1, \ldots, \beta_{\dim(A_0)})$.

Example 2.1. (1) Let $G_1 = (N_1, T_1, F_1, P_1, S)$ be an MCFG where $N_1 = \{S, A\}$, $T_1 = \{a, b\}$ and $P_1 = \{S \to J[A],\ A \to f_a[A] \mid f_b[A] \mid (\varepsilon, \varepsilon)\}$ where $\dim(S) = 1$, $\dim(A) = 2$, $J[(x_1, x_2)] = x_1 x_2$ and $f_\alpha[(x_1, x_2)] = (\alpha x_1, \alpha x_2)$ with $\alpha = a, b$.

(2) Let $G_2 = (N_2, T_2, F_2, P_2, S)$ be an MCFG where $N_2 = \{S, A\}$, $T_2 = \{a_i \mid 1 \leq i \leq 2m\}$ and $P_2 = \{S \to J_m[A],\ A \to g[A] \mid (\varepsilon, \ldots, \varepsilon)\}$ where $\dim(S) = 1$, $\dim(A) = m$, $J_m[(x_1, \ldots, x_m)] = x_1 \cdots x_m$ and $g[(x_1, \ldots, x_m)] = (a_1 x_1 a_2, \ldots, a_{2m-1} x_m a_{2m})$.

(3) Let $G_3 = (N_2, T_2, F_3, P_3, S)$ be an MCFG where $P_3 = \{S \to J_m^2[A, A],\ A \to g[A] \mid (\varepsilon, \ldots, \varepsilon)\}$ where $J_m^2[(x_1, \ldots, x_m), (y_1, \ldots, y_m)] = x_1 \cdots x_m y_1 \cdots y_m$. □

For a function f defined by (2.1) in condition (F) and tuples of terminal sequences $\overline{\alpha_i} = (\alpha_{i1}, \ldots, \alpha_{id_i}) \in (T^*)^{d_i}$ $(1 \leq i \leq k)$, let $f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ denote the tuple of terminal sequences obtained from the right hand sides of (2.1) by substituting α_{ij} $(1 \leq i \leq k,\ 1 \leq j \leq d_i)$ into x_{ij}. For instance, $f_a[(bba, ab)] = (abba, aab)$ in Example 2.1 (1). We recursively define the relation $\overset{*}{\Rightarrow}_G$ by the following (L1) and (L2):

(L1) If $A \to \overline{\alpha} \in P$ $(\overline{\alpha} \in (T^*)^{\dim(A)})$, we write $A \overset{*}{\Rightarrow}_G \overline{\alpha}$.

(L2) If $A \to f[A_1, \ldots, A_k] \in P$ and $A_i \overset{*}{\Rightarrow}_G \overline{\alpha_i}$ $(1 \leq i \leq k)$, we write $A \overset{*}{\Rightarrow}_G f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$.

We will omit the subscript G if it is clear from the context. For $A \in N$, the set generated from A in G is defined as $L_G(A) = \{\overline{w} \in (T^*)^{\dim(A)} \mid A \overset{*}{\Rightarrow}_G \overline{w}\}$ and the language generated by G is defined as $L(G) = L_G(S)$. A language L is a *multiple context-free language* (MCFL) if there exists an MCFG G such that $L = L(G)$. The class of all MCFGs and the class of all MCFLs are denoted by $\mathcal{MCFG}$ and $\mathcal{MCFL}$ respectively. The same notational convention will be used for other classes of grammars and languages. In parallel with the relation $\overset{*}{\Rightarrow}_G$, we define derivation trees:

(D1) If $A \to \overline{\alpha} \in P$ $(\overline{\alpha} \in (T^*)^{\dim(A)})$, a derivation tree for $\overline{\alpha}$ is the tree with a single node labeled $A : \overline{\alpha}$.

(D2) If $A \to f[A_1, \ldots, A_k] \in P$, $A_i \overset{*}{\Rightarrow}_G \overline{\alpha_i}$ $(1 \leq i \leq k)$ and $t_1, \ldots, t_k$ are derivation trees for $\overline{\alpha_1}, \ldots, \overline{\alpha_k}$, then a derivation tree for $f[\overline{\alpha_1}, \ldots, \overline{\alpha_k}]$ is the tree with the root labeled $A : f$ that has $t_1, \ldots, t_k$ as (immediate) subtrees from left to right.

Example 2.1 (continued)**.** (1) By (L1), $A \overset{*}{\Rightarrow}_{G_1} (\varepsilon, \varepsilon)$ since $A \to (\varepsilon, \varepsilon) \in P$. Since $f_a[(\varepsilon, \varepsilon)] = (a, a)$ and $f_b[(a, a)] = (ba, ba)$, we have $A \overset{*}{\Rightarrow}_{G_1} (a, a)$ and $A \overset{*}{\Rightarrow}_{G_1} (ba, ba)$ by (L2). Also by $S \to J[A]$, $S \overset{*}{\Rightarrow}_{G_1} J[(ba, ba)] = baba$. In fact, $L_{G_1}(A) = \{(w, w) \mid w \in \{a, b\}^*\}$ and $L(G_1) = \{ww \mid w \in \{a, b\}^*\}$.

(2) Likewise, $A \overset{*}{\Rightarrow}_{G_2} (\varepsilon, \ldots, \varepsilon)$ by (L1), $A \overset{*}{\Rightarrow}_{G_2} f[(\varepsilon, \ldots, \varepsilon)] = (a_1 a_2, \ldots, a_{2m-1} a_{2m})$ by (L2), etc. This tells us that $L(G_2) = \{a_1^n \cdots a_{2m}^n \mid n \geq 0\}$.

(3) Since $L_{G_3}(A) = L_{G_2}(A)$, $L(G_3) = \{\alpha\beta \mid \alpha, \beta \in L(G_2)\}$. □

To introduce subclasses of $\mathcal{MCFG}$, we define a few terminologies. Let $G = (N, T, F, P, S)$ be an arbitrary MCFG. For a function $f : (T^*)^{d_1} \times \cdots \times (T^*)^{d_k} \to (T^*)^{d_0}$, let $\dim(f) = d_0$, $\text{rank}(f) = k$ and $\deg(f) = \sum_{j=0}^{k} d_j$, which are called the *dimension*, *rank* and *degree* of f respectively. $\dim(G)$, $\text{rank}(G)$ and $\deg(G)$ are defined as the maximum of $\dim(f)$, $\text{rank}(f)$ and $\deg(f)$ among all $f \in F$ respectively. By definition, $\deg(G) \leq \dim(G)(\text{rank}(G) + 1)$. With these parameters, we define subclasses of $\mathcal{MCFG}$. An MCFG G with $\dim(G) \leq m$ and $\text{rank}(G) \leq r$ is called an (m, r)-MCFG. Likewise, an MCFG G with $\dim(G) \leq m$ is called an m-MCFG.

The following proposition was shown by Rambow and Satta, which summarizes Theorems 1 and 6 of [21, 22].

Proposition 2.1 ([21, 22])**.** For $m \geq 2,\ r \geq 6$,
$L_{r,m} \in (m, r-2)\text{-}\mathcal{MCFL} \setminus (m, r-3)\text{-}\mathcal{MCFL}$. □

In case $m = 2$ and $r = 6$, $L_{6,2} \in (2, 4)\text{-}\mathcal{MCFL} \setminus (2, 3)\text{-}\mathcal{MCFL}$. It was also shown in [21, 22] that $(2, 2)\text{-}\mathcal{MCFL} = (2, 3)\text{-}\mathcal{MCFL}$. Therefore, $L_{6,2} \in (2, 4)\text{-}\mathcal{MCFL} \setminus (2, 2)\text{-}\mathcal{MCFL}$, which implies $(2, 2)\text{-}\mathcal{MCFL} \subsetneq 2\text{-}\mathcal{MCFL}$.

Example 2.2. Consider a $(2, 2)$-MCFG $G_4 = (\{S, A\}, \{a, c, g, u\}, F_4, P_4, S)$ for generating RNA sequences where P_4 and F_4 are shown in Table 2.1. Functions have mnemonic names where XS, BF, BP and UP stand for CROSSING, BIFURCATION, BASE PAIR and UNPAIR respectively. The RNA sequence agacuu in Figure 2.2 can be generated by the above rules as follows: $A \overset{*}{\Rightarrow}_{G_4} BP^{gc}[(\varepsilon, \varepsilon)] = (g, c)$, $A \overset{*}{\Rightarrow}_{G_4} BP^{au}[(g, c)] = (ag, cu)$, $A \overset{*}{\Rightarrow}_{G_4} BP^{au}[(\varepsilon, \varepsilon)] = (a, u)$, $A \overset{*}{\Rightarrow}_{G_4} XS_2[(ag, cu), (a, u)] = (aga, cuu)$ and $S \overset{*}{\Rightarrow}_{G_4} J[(aga, cuu)] = agacuu$. G_4 has a derivation tree (Figure 2.3) for agacuu that represents the pseudoknot shown in Figure 2.2. □

Recognition problems for MCFGs can be solved in polynomial time:

Proposition 2.2 ([15, 29])**.** Let G be an MCFG with $\deg(G) = e$. For a given $w \in T^*$, whether $w \in L(G)$ or not can be decided in $O(n^e)$ time where $n = |w|$. □

Table 2.1. MCFG G_4

Rule	Function
$S \to J[A]$	$J[(x_1, x_2)] = x_1 x_2$
$A \to XS_1[A, A]$	$XS_1[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}, x_{21} x_{12} x_{22})$
$A \to XS_2[A, A]$	$XS_2[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11} x_{21}, x_{12} x_{22})$
$A \to XS_3[A, A]$	$XS_3[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11} x_{21} x_{12}, x_{22})$
$A \to BF_1[A, A]$	$BF_1[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}, x_{12} x_{21} x_{22})$
$A \to BF_2[A, A]$	$BF_2[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11} x_{12}, x_{21} x_{22})$
$A \to BF_3[A, A]$	$BF_3[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11} x_{12} x_{21}, x_{22})$
$A \to UP^{\alpha}_{1L}[A]$	$UP^{\alpha}_{1L}[(x_1, x_2)] = (\alpha x_1, x_2)$
$A \to UP^{\alpha}_{1R}[A]$	$UP^{\alpha}_{1R}[(x_1, x_2)] = (x_1 \alpha, x_2)$
$A \to UP^{\alpha}_{2L}[A]$	$UP^{\alpha}_{2L}[(x_1, x_2)] = (x_1, \alpha x_2)$
$A \to UP^{\alpha}_{2R}[A]$	$UP^{\alpha}_{2R}[(x_1, x_2)] = (x_1, x_2 \alpha)$
$A \to BP^{\alpha\beta}[A]$	$BP^{\alpha\beta}[(x_1, x_2)] = (\alpha x_1, x_2 \beta)$
$A \to (\varepsilon, \varepsilon)$	

Note: $\alpha \in \{a, c, g, u\}$ and $(\alpha, \beta) \in \{(a, u), (u, a), (c, g), (g, c)\}$.

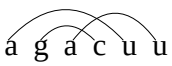


Figure 2.2. Example of a pseudoknot

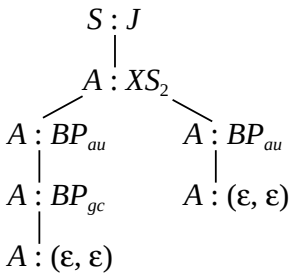


Figure 2.3. A derivation tree in G_4

Chapter 3

Generative Power of Grammars for RNA Pseudoknotted Structure

1. Introduction

Much attention has been paid to RNA secondary structure prediction techniques based on context-free grammars (CFGs) since CFGs can represent stem loop structures (Figure 1.1 (a)) by their derivation trees and parsing (or secondary structure prediction in biological words) can be performed in $O(n^3)$ time where n is the length of an input RNA sequence. A pseudoknot (Figure 1.1 (b)) is one of the typical substructures found in RNA secondary structures. An alternative representation of a pseudoknot is arc depiction in which arcs cross (see Figure 1.2 (b)). It has been recognized that pseudoknots play an important role in RNA functions such as ribosomal frameshifting and splicing. Also, a database (PseudoBase [4]) containing a variety of structural, functional and sequence data on RNA pseudoknots has been constructed. However, it is known that CFGs cannot represent pseudoknotted structure [1].

Since the middle 1980s, there has been a significant concept for syntax of natural language, which is called mildly context-sensitive grammar (mildly CSG). The common features of these grammars are generative power between CFGs and CSGs, and polynomial time recognizability. Kasami et al. [14, 15, 29] proposed a class of gram-

[1]Formal grammars are also used for modeling other functions on molecular sequences with secondary structures. For example, Sakakibara and Ferretti [27] showed that splicing systems on trees can generate context-free languages.

mars called multiple context-free grammars (MCFGs), which are natural extension of CFGs. A nonterminal symbol of an MCFG derives tuples of sequences, while a nonterminal symbol of a CFG derives sequences. Tree adjoining grammars (TAGs) [11, 12] and MCFGs have been known as typical instances of mildly CSGs [13].

In bioinformatics, a few grammars have been proposed to represent pseudoknots. In one pioneering paper [30], Uemura et al. defined two subclasses of TAGs called simple linear TAGs (SLTAGs) and extended SLTAGs (ESLTAGs). Rivas and Eddy [24] introduced a new class of grammars called RNA pseudoknot grammars (RPGs) for deriving sequences with gaps. These grammars have generative power stronger than CFGs, while recognition can be performed in polynomial time. However, the relation between the generative power of each of these grammars has not been clarified.

In this chapter, we identify these grammars for describing RNA secondary structure as subclasses of MCFGs, and clarify the inclusion relation between the classes of languages generated by these grammars. The remainder of this chapter is organized as follows. First, we review the grammars for describing RNA pseudoknotted structure mentioned above in Section 2. In Section 3, these grammars are characterized as subclasses of MCFGs. The closure property and generative power of these grammars are investigated in Section 4 and Section 5 respectively. Section 6 concludes this chapter.

2. Grammars for Describing RNA Pseudoknotted Structure

2.1 Tree Adjoining Grammar

Basic Definitions

We first define notation for trees. Let $\mathbb{N}$ be the set of positive integers. Then the partial order $\preceq$ over $\mathbb{N}^*$ is defined as follows: $p \preceq q$ for $p, q \in \mathbb{N}^*$ if and only if there exists $r \in \mathbb{N}^*$ such that $q = pr$. We write $p \prec q$ when $p \preceq q$ and $p \neq q$. Let Σ be a finite set of symbols. A tree t over $\Sigma \cup \{\varepsilon\}$ is defined as a function such that $t : D_t \rightarrow \Sigma \cup \{\varepsilon\}$ where D_t is a finite subset of $\mathbb{N}^*$ satisfying the following conditions:

(1) If $q \in D_t$ and $p \prec q$, then $p \in D_t$.

(2) If $pj \in D_t$ and $j \in \mathbb{N}$, then $p1, \ldots, p(j-1) \in D_t$.

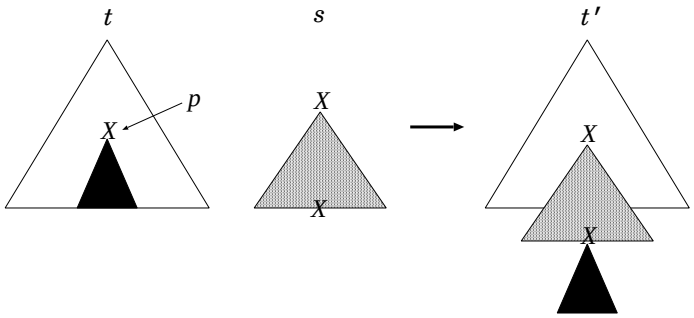


Figure 3.1. Adjoining operation

Each element in D_t is called an *address*. The address $p = \varepsilon$ especially indicates the root node of t. If $p \prec q$ does not hold for any $q \in D_t$, then p indicates a leaf node. We say that $t(p)$ is the label of the node at address p in the tree t. The *yield* of a tree t (denoted by yield(t)) is the sequence obtained by concatenating the labels of leaf nodes of t from left to right.

A *tree adjoining grammar* (TAG) is a 5-tuple $G = (N, T, S, \mathcal{I}, \mathcal{A})$ where N and T are finite sets of nonterminals and terminals, respectively, $S \in N$ is the start symbol, $\mathcal{I}$ is a finite set of *initial trees* (*center trees*) over $N \cup T \cup \{\varepsilon\}$ and $\mathcal{A}$ is a finite set of *adjunct trees* (*auxiliary trees*) over $N \cup T \cup \{\varepsilon\}$. $\mathcal{I}$ and $\mathcal{A}$ satisfy the following conditions:

(1) If $t_1 \in \mathcal{I}$, then $t_1(\varepsilon) = S$ and yield$(t_1) \in T^*$.

(2) If $t_2 \in \mathcal{A}$, then $t_2(\varepsilon) = X$ and yield$(t_2) \in T^*XT^*$ for some $X \in N$.

The leaf node whose label is the same as the label of the root node of an adjunct tree is called the *foot node*. The path of an adjunct tree from the root node to the foot node is called the *backbone*. All initial and adjunct trees are referred to as *elementary trees*.

We next define the adjoining operation over trees. Let t be a tree with the node labeled X at address p. Let s be an adjunct tree with root and foot labeled X. Then we say that s is adjoinable to t at p, and the tree t' obtained from t by adjoining s at p is defined as shown in Figure 3.1. Also, we write $t \vdash_s t'$ (or simply $t \vdash t'$). We write the reflective and transitive closure of $\vdash$ as $\vdash^*$. We call t' a *derived tree* (or a tree derived from t) if $t \vdash^* t'$ for some $t \in \mathcal{I} \cup \mathcal{A}$.

An *adjoining constraint* for a node n of an elementary tree is as follows:

(1) *Selective Adjoining* (SA($\mathcal{T}$)) where $\mathcal{T} \subseteq \mathcal{A}$ $(\mathcal{T} \neq \phi)$: Only members of $\mathcal{T}$ can be adjoined at n.

(2) *Null Adjoining* (NA): No adjunct tree can be adjoined at n.

(3) *Obligatory Adjoining* (OA($\mathcal{T}$)) where $\mathcal{T} \subseteq \mathcal{A}$ $(\mathcal{T} \neq \phi)$: A member of $\mathcal{T}$ must be adjoined at n.

If a node has none of the three constraints mentioned above, we interpret its constraint as SA($\mathcal{A}$). Therefore, we assume that every node has exactly one of the three adjoining constraints. The relation $t \vdash_s t'$ (and adjoinability) are redefined so that t' is obtained from t by adjoining s at p (of node n) where n has no NA and if n has SA($\mathcal{T}$), then $s \in \mathcal{T}$. A node n is *inactive* if the constraint for the node is NA, otherwise *active*. If no active node in a tree t has OA constraint, then t is called *mature*.

The tree set of a TAG G is defined as $T(G) = \{t \mid s \vdash^* t,\ s \in \mathcal{I}$ and t is mature$\}$. The definition of the relation $\vdash^*$ is top down in the sense that only an adjunct tree can be adjoined to a derived tree. As discussed in [32], $T(G)$ can be alternatively characterized in a bottom up way by allowing derived trees to be adjoined to a tree. For example, if $s_0 \vdash_{s_1} t_1 \vdash_{s_2} t_2$ where $s_0 \in \mathcal{I}$, $s_1, s_2 \in \mathcal{A}$ and s_2 is adjoined at a node inherited from s_1, then we can first adjoin s_2 to s_1 resulting in τ and then adjoin τ to s_0 to obtain t_2. Note that in this bottom up definition, we can restrict a tree to which a derived tree is adjoined to be an elementary tree (like s_1 and s_0 above). For each $s \in \mathcal{I} \cup \mathcal{A}$, let us define a series of tree sets $T_0^s(G),\ T_1^s(G), \ldots$.

(T1) $T_0^s(G) = \{s\}$ if s is mature and $T_0^s(G) = \phi$ otherwise.

(T2) $T_{n+1}^s(G) = T_n^s(G) \cup \{\tau \mid s \vdash_{\sigma_1} \tau_1 \vdash_{\sigma_2} \cdots \vdash_{\sigma_k} \tau_k = \tau,\ \sigma_i \in T_n(G)\ (1 \leq i \leq k),\ p_1, \ldots, p_k$ are different addresses of $s,\ \sigma_i$ is adjoinable to s at $p_i\ (1 \leq i \leq k)$ and τ is mature$\}$.

(T3) $T_n(G) = \bigcup_{s \in \mathcal{I} \cup \mathcal{A}} T_n^s(G)$ for each $n \geq 0$.

It is not difficult to show that $T(G) = \{t \mid t \in T_n(G)$ for some $n \geq 0$ and yield$(t) \in T^*\}$. This characterization of $T(G)$ by (T1) through (T3) is frequently used in the proofs in Section 3.

The language generated by G is defined as $L(G) = \{w \mid w =$ yield$(t),\ t \in T(G)\}$, which is called a *tree adjoining language* (TAL). As in MCFG, Let $\mathcal{TAG}$ denote the

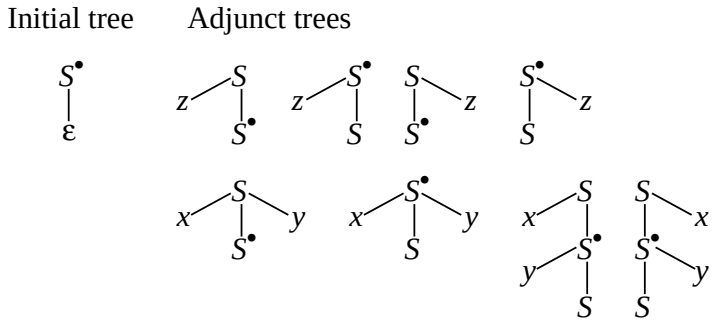


Figure 3.2. Elementary trees in Example 3.1

class of TAGs and $\mathcal{TAL}$ denote the class of TALs. It has been proved that

$$\mathcal{TAL} \subsetneq (2,2)\text{-}\mathcal{MCFL} \subsetneq 2\text{-}\mathcal{MCFL} \subsetneq \mathcal{MCFL} \tag{3.1}$$

where the proper inclusion relations of the leftmost and the rightmost in (3.1) were given by Lemma 4.15 of [29] and Lemma 5 of [14] respectively.

SLTAG and ESLTAG

We now define *simple linear TAGs* (SLTAGs) and *extended simple linear TAGs* (ESLTAGs) introduced in [30]. An elementary tree is *simple linear* if it has exactly one active node, and for an adjunct tree, the active node is on the backbone of the tree. A TAG G is a *simple linear TAG* (SLTAG) if and only if all elementary trees in G are simple linear. An adjunct tree is *semi-simple linear* if it has two active nodes, where one is on the backbone and the other is elsewhere. A TAG G is an *extended simple linear TAG* (ESLTAG) if and only if all initial trees in G are simple linear and all adjunct trees in G are either simple linear or semi-simple linear.

Example 3.1 ([30])**.** Let $G_1 = (N_1, T_1, S, \mathcal{I}_1, \mathcal{A}_1)$ be an SLTAG where $N_1 = \{S\}$, $T_1 = \{a, c, g, u\}$ and elementary trees in $\mathcal{I}_1$ and $\mathcal{A}_1$ are shown in Figure 3.2. In the figure, $z \in \{a, c, g, u\}$, $(x, y) \in \{(a, u), (u, a), (c, g), (g, c)\}$ and an active node is denoted by $S^\bullet$. Figure 3.3 shows a (top down) derivation of a pseudoknot. Figure 3.4 shows an example of a bottom up derivation where a shaded region is a derived tree adjoined to an elementary tree. □

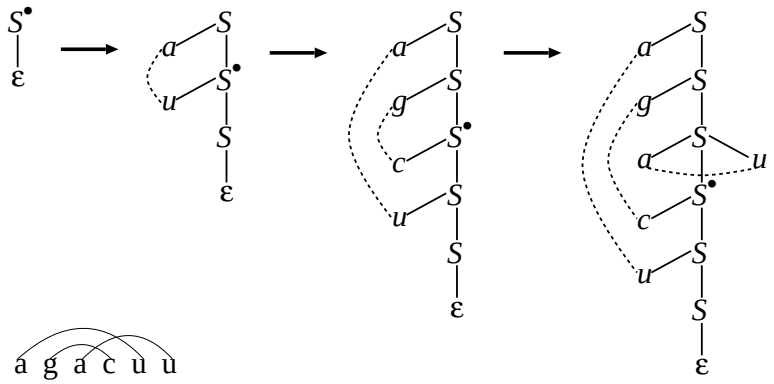


Figure 3.3. A derivation of a pseudoknot in Example 3.1

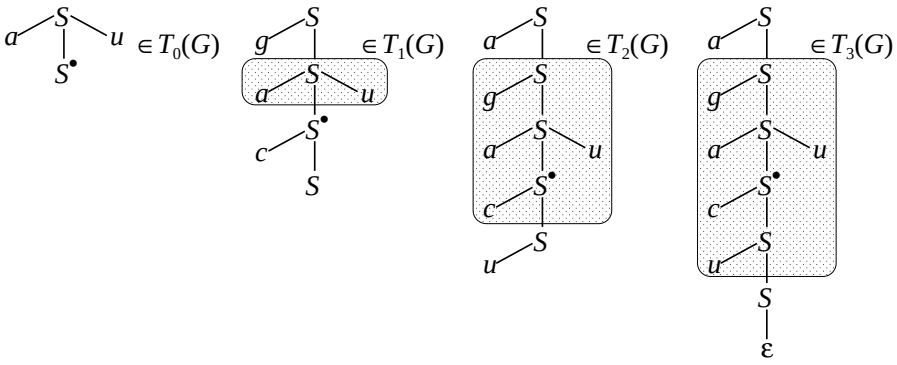


Figure 3.4. A bottom up derivation in Example 3.1

Example 3.2. Let $G_2 = (N_2, T_2, S, \mathcal{I}_2, \mathcal{A}_2)$ be an ESLTAG where $N_2 = \{S, A_i \mid 1 \leq i \leq 6\}$, $T_2 = \{\sharp, a_i, b_i, c_i \mid 1 \leq i \leq 4\}$ and elementary trees in $\mathcal{I}_2$ and $\mathcal{A}_2$ are shown in Figure 3.5. G_2 generates $L_3 = \{\sharp a_1^k b_1^k c_1^k \sharp a_2^l b_2^l c_2^l \sharp a_3^m b_3^m c_3^m \sharp a_4^n b_4^n c_4^n \sharp \mid k, l, m, n \geq 1\}$. □

By definition,

$$\mathcal{SLTAL} \subseteq \mathcal{ESLTAL} \subseteq \mathcal{TAL}. \tag{3.2}$$

On the inclusion relation between $\mathcal{CFL}$, $\mathcal{SLTAL}$ and $\mathcal{ESLTAL}$, the following has

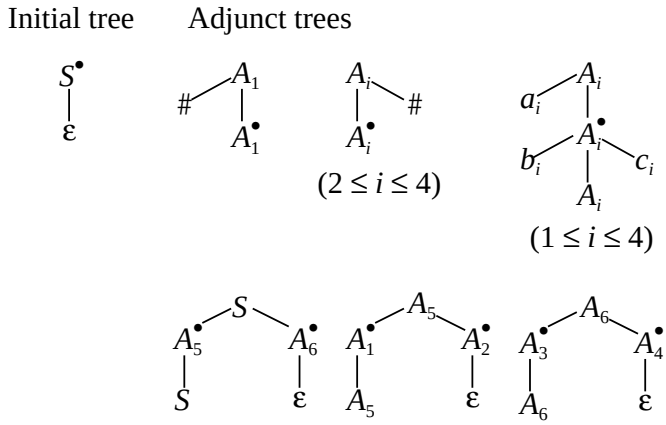


Figure 3.5. Elementary trees in Example 3.2

been shown in Propositions 1 to 3 of [30]:

$$L_2 = \{\sharp a_1^k b_1^k \sharp a_2^l b_2^l \sharp a_3^m b_3^m \sharp a_4^n b_4^n \sharp \mid k, l, m, n \ge 1\} \in \mathcal{CFL} \setminus \mathcal{SLTAL}, \tag{3.3}$$

$$\{a^n b^n c^n \mid n \ge 0\} \in \mathcal{SLTAL} \setminus \mathcal{CFL}, \tag{3.4}$$

$$\mathcal{CFL} \subseteq \mathcal{ESLTAL}. \tag{3.5}$$

Satta and Schuler's Subclass

From the viewpoint of parsing complexity, Satta and Schuler [28] proposed a subclass of $\mathcal{TAG}$, which we call $\mathcal{SSTAG}$. They classified adjunct trees into three types according to the position of the backbone. Each of them is called a *left tree*, a *right tree* and a *wrapping tree* and satisfies the following (LT1), (RT1) and (WT1) respectively:

(LT1) The rightmost leaf is the foot node and the backbone consists of only the root and the foot nodes.

(RT1) The leftmost leaf is the foot node and the backbone consists of only the root and the foot nodes.

(WT1) Neither (LT1) nor (RT1) holds.

A TAG G is called an *SSTAG* if and only if each left tree, right tree and wrapping tree in G satisfies the following (LT2), (RT2) and (WT2) respectively:

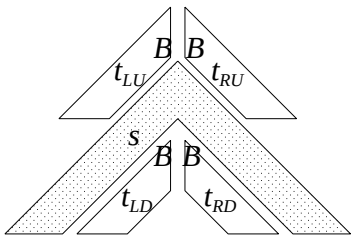


Figure 3.6. Wrapping adjunction

(LT2) At the backbone of each left tree, no wrapping tree can be adjoined and no adjoining constraint on right tree is found.

(RT2) At the backbone of each right tree, no wrapping tree can be adjoined and no adjoining constraint on left tree is found.

(WT2) At the backbone of each wrapping tree, there is at most one node where a wrapping tree can be adjoined, which is called the *wrapping node*.

We describe the reason why the above restriction is imposed. The most time-consuming step in the recognition on TAG is the one dealing with adjoining operation. When we adjoin a wrapping tree s to the wrapping node B of another wrapping tree t, we can split t at B into four parts $t_{LU}, t_{LD}, t_{RU}, t_{RD}$ and adjunction can be simulated by four successive steps (see Figure 3.6). Specifically, s is combined with t_{LD} resulting σ_1 and then σ_1 is combined with t_{RD} resulting σ_2, etc. In the recognition on SSTAGs, every wrapping tree is split into four adjunct trees and regarded as a left (or right) tree satisfying (LT2) (or (RT2)). This plays an important role in the reduction of the time complexity. In the following, we assume that the wrapping node differs from the root and the foot nodes without loss of generality. The close relation between ESLTAGs and SSTAGs will be investigated in Section 5.

2.2 RNA Pseudoknot Grammar

Rivas and Eddy [24] introduced *crossed-interaction grammars* (CIGs), which are similar to MCFGs. A CIG has a special terminal symbol $\wedge$ (called the “hole” or the “gap”) and some symbols (called *extra nonterminal symbols*) other than terminals and nonterminals. An extra nonterminal symbol plays a similar role to a function in MCFGs,

and the semantics of an extra nonterminal is given by a rearrangement rule. The hole $\wedge$ provides the insertion position in a rearrangement rule. Rivas and Eddy defined a subclass of $\mathcal{CIG}$ to describe RNA secondary structure including pseudoknots. In the following, these grammars are briefly reviewed.

A *crossed-interaction grammar* (CIG) is a 6-tuple $G = (N, T, S, I, P, R)$ where N is a finite set of nonterminal symbols, T is a finite set of terminal symbols that contains a distinguished symbol $\wedge$ called the hole (or the gap), $S \in N$ is the start symbol, I is a finite set of extra nonterminal symbols, P is a finite set of production rules (productions) and R is a countable set of rearrangement rules (rearrangements). A production is of the form $A \to \alpha\ (A \in N,\ \alpha \in (N(IN)^* \cup T)^*)$ and a rearrangement is of the form $(\beta) \to_R m\ (\beta \in (T \cup I)^*,\ m \in T^*)$. For $\gamma, \delta \in (N \cup T \cup I)^*$, we write $\gamma A \delta \Rightarrow_G \gamma \alpha \delta$ if $A \to \alpha \in P$, and $\gamma(\beta)\delta \Rightarrow_{G,R} \gamma m \delta$ if $(\beta) \to_R m \in R$. The reflective and transitive closure of $\Rightarrow_G$ and $\Rightarrow_{G,R}$ are denoted as $\overset{*}{\Rightarrow}_G$ and $\overset{*}{\Rightarrow}_{G,R}$ respectively. The subscript G is omitted if it is clear from the context. The language generated by G is defined as $L(G) = \{h_\wedge(w) \mid S \overset{*}{\Rightarrow}_G \gamma \overset{*}{\Rightarrow}_{G,R} w,\ \gamma \in (T \cup I)^*,\ w \in T^*\}$ where $h_\wedge$ is the homomorphism such that $h_\wedge(\wedge) = \varepsilon$ and $h_\wedge(x) = x$ for $x \in T \setminus \{\wedge\}$.

An *RNA pseudoknot grammar* (RPG) is a CIG $G = (N, T, S, I, P, R)$ where I is fixed to $\{\times_R, \times, \times_L, \supset, (,)\}$ and R is fixed to

$$
\begin{aligned}
&(u_1 \wedge u_2 \times_R v_1 \wedge v_2) \to_R u_1 \wedge v_1 u_2 v_2,\\
&(u_1 \wedge u_2 \times v_1 \wedge v_2) \to_R u_1 v_1 \wedge u_2 v_2,\\
&(u_1 \wedge u_2 \times_L v_1 \wedge v_2) \to_R u_1 v_1 u_2 \wedge v_2,\\
&(u_1 \wedge u_2 \supset v_1 \wedge v_2) \to_R u_1 v_1 \wedge v_2 u_2
\end{aligned}
$$

for each $u_1, u_2, v_1, v_2 \in T^*$. Since I and R are fixed, we will write an RPG as $G = (N, T, S, P)$.

Example 3.3. Let $G = (N, T, S, P)$ be an RPG where $N = \{S, A\},\ T = \{a, b\}$ and $P = \{S \to (A \times A) \mid \varepsilon,\ A \to (A \times A) \mid a \wedge a \mid b \wedge b \mid \wedge\}$. Then $S \Rightarrow (A \times A) \Rightarrow (A \times (A \times A)) \overset{*}{\Rightarrow} (a \wedge a \times (b \wedge b \times a \wedge a)) \Rightarrow_R (a \wedge a \times ba \wedge ba) \Rightarrow_R aba \wedge aba$. Thus, $h_\wedge(aba \wedge aba) = abaaba \in L(G)$. In fact, $L(G) = \{ww \mid w \in \{a, b\}^*\}$. $\square$

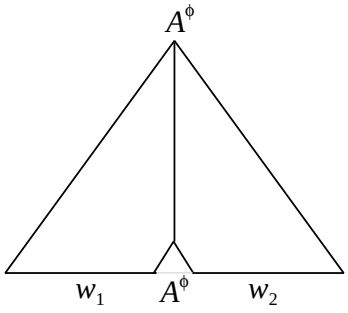


Figure 3.7. A mature derived tree

3. New Subclasses of $\mathcal{MCFG}$

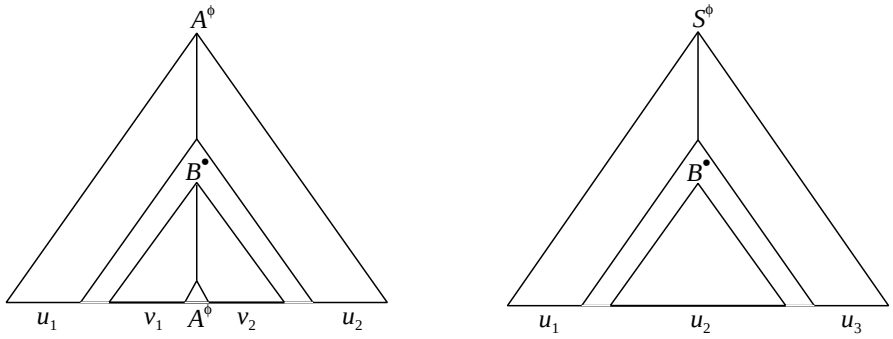
3.1 A Subclass of $\mathcal{MCFG}$ for $\mathcal{SLTAL}$

Grammars G and G' are called weakly equivalent if $L(G) = L(G')$. In [31], the following translation method from a TAG $G = (N, T, S, \mathcal{I}, \mathcal{A})$ into a weekly equivalent $(2, 2)$-MCFG G' has been proposed: For each nonterminal $A \in N$ in G, a nonterminal A with $\dim(A) = 2$ is introduced in G', and rules are constructed so that there exists a mature derived tree t such that yield$(t) = w_1 A w_2$ $(w_1, w_2 \in T^*)$ in G (Figure 3.7) if and only if $A \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$. Remember that each elementary tree in an SLTAG contains exactly one active node as shown in Figure 3.8 (An inactive node and an active node are denoted like A^{ϕ} and $B^{\bullet}$, respectively, in the figure). By utilizing this restriction, we can define a translation for SLTAG simpler than that of [31]. Namely, for an adjunct tree in Figure 3.8 (a), construct an MCFG rule $A \rightarrow f[B]$ where $f[(x_1, x_2)] = (u_1 x_1 v_1, v_2 x_2 u_2)$. This translation motivates us to define the following subclass of $(2, 1)$-$\mathcal{MCFG}$.

Definition 3.1. A $(2, 1)$-MCFG $G = (N, T, F, P, S_0)$ is an SLMCFG if G satisfies the following conditions (1) and (2):

(1) For each nonterminal A other than S_0, $\dim(A) = 2$.

(2) Each nonterminating rule has the form of either $S_0 \rightarrow J[A]$ where $J[(x_1, x_2)] = x_1 x_2$ or $A \rightarrow f[B]$ where $A, B \in N \setminus \{S_0\}$ and $f[(x_1, x_2)] = (u_1 x_1 v_1, v_2 x_2 u_2)$ for some $u_j, v_j \in T^*$ $(j = 1, 2)$. Such a function f is called a *simple linear function.* □



(a) Adjunct tree (b) Initial tree

Figure 3.8. Elementary trees in SLTAG

In the next lemma, we show that the generative power of $\mathcal{SLMCFG}$ is the same as that of $\mathcal{SLTAG}$. In the beginning of this section, we already mentioned an idea of translating from an SLTAG G into a weakly equivalent SLMCFG G'. Considering SA constraint, we introduce a nonterminal $A^{[t]}$ into G' for a nonterminal A and an elementary tree t in G. $A^{[t]}$ is intended to derive (w_1, w_2) if and only if there exists a mature derived tree τ such that yield$(\tau) = w_1 A w_2$ and τ is obtained from t by adjoining derived trees to t.

Lemma 3.1. $\mathcal{SLTAL} = \mathcal{SLMCFL}$.

Proof. $(\mathcal{SLTAL} \subseteq \mathcal{SLMCFL})$ Let $G = (N, T, S, \mathcal{I}, \mathcal{A})$ be a given SLTAG. We will construct an SLMCFG $G' = (N', T, F, P, S_0)$ as follows:

(1) $N' = \{A^{[t]} \mid A \in N,\ t \in \mathcal{I} \cup \mathcal{A}\} \cup \{S_0\}$ where $\dim(S_0) = 1$ and $\dim(A^{[t]}) = 2$ for each $A \in N$ and $t \in \mathcal{I} \cup \mathcal{A}$.

(2) P (and F) are the smallest sets satisfying the following conditions (a) through (c).

(a) $S_0 \to J[S^{[t]}] \in P$ for each $t \in \mathcal{I}$ and $J[(x_1, x_2)] = x_1 x_2 \in F$.

(b) For each adjunct tree $t \in \mathcal{A}$ shown in Figure 3.8 (a),

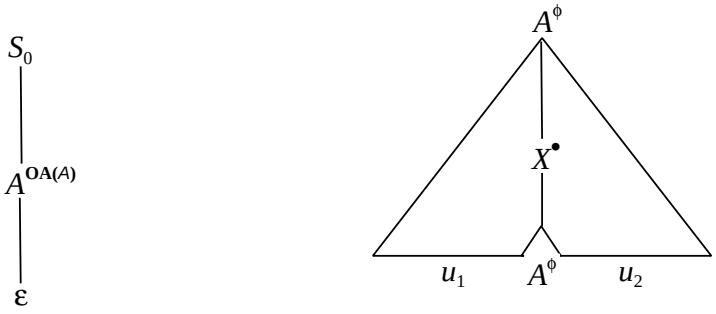
- $A^{[t]} \to f[B^{[s]}] \in P$ for each $s \in \mathcal{T}$ and $f[(x_1, x_2)] = (u_1 x_1 v_1, v_2 x_2 u_2) \in F$ if B has either SA($\mathcal{T}$) or OA($\mathcal{T}$), and
- $A^{[t]} \to (u_1 v_1, v_2 u_2)$ if B has either SA($\mathcal{T}$) or NA (i.e., t is mature).

(c) For each initial tree $t \in \mathcal{I}$ shown in Figure 3.8 (b),

- $S^{[t]} \rightarrow g[B^{[s]}] \in P$ for each $s \in \mathcal{T}$ and $g[(x_1, x_2)] = (u_1x_1u_2, x_2u_3) \in F$ if B has either SA($\mathcal{T}$) or OA($\mathcal{T}$), and
- $S^{[t]} \rightarrow (u_1u_2, u_3)$ if t is mature.

First, we show that there exists a tree $\tau \in T_n^t(G)$ for some $n \geq 0$ such that yield$(\tau) = w_1Aw_2$ $(A \in N,\ w_1, w_2 \in T^*)$ if and only if $A^{[t]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$.
("if" part) By induction on the application number of (L1) and (L2) in Section 2 of Chapter 2.
(Basis) If $A^{[t]} \Rightarrow_{G'} (w_1, w_2)$, that is, $A^{[t]} \rightarrow (w_1, w_2) \in P$, there should exist an adjunct tree t that corresponds to the rule by construction (2) (b) (see Figure 3.9 (b)). Then yield$(t) = w_1Aw_2$.
(Induction) Assume that $B^{[s]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$ and $A^{[t]} \stackrel{*}{\Rightarrow}_{G'} f[(w_1, w_2)] = (u_1w_1v_1, v_2w_2u_2)$ by $A^{[t]} \rightarrow f[B^{[s]}] \in P$ where $f[(x_1, x_2)] = (u_1x_1v_1, v_2x_2u_2)$. Then there should exist an adjunct tree t shown in Figure 3.8 (a) that corresponds to $A^{[t]} \rightarrow f[B^{[s]}]$ by construction (2) (b). By the inductive hypothesis, there exists a mature derived tree $\sigma \in T_n^s(G)$ such that yield$(\sigma) = w_1Bw_2$. Therefore, $t \vdash_\sigma \tau$ where yield$(\tau) = u_1w_1v_1Av_2w_2u_2$.
("only if" part) By induction on n.
(Basis) If $n = 0$, that is, there exists a mature adjunct tree $\tau = t$ such that yield$(\tau) = w_1Aw_2$, then a rule $A^{[t]} \rightarrow (w_1, w_2)$ is constructed by construction (2) (b). Thus, $A^{[t]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$ by (L1).
(Induction) Assume that a mature tree $\tau \in T_n^t(G)$ is obtained by adjoining a tree $\sigma \in T_{n-1}^s(G)$ such that yield$(\sigma) = w_1Bw_2$ to an adjunct tree t shown in Figure 3.8 (a). Then yield$(\tau) = u_1w_1v_1Av_2w_2u_2$. By construction (2) (b), $A^{[t]} \rightarrow f[B^{[s]}] \in P$ where $f[(x_1, x_2)] = (u_1x_1v_1, v_2x_2u_2)$. By the inductive hypothesis, $B^{[s]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$. Hence, $A^{[t]} \stackrel{*}{\Rightarrow}_{G'} f[(w_1, w_2)] = (u_1w_1v_1, v_2w_2u_2)$ by (L2).

Next, we show that $L(G) = L(G')$ by considering the correspondence between the initial trees in $\mathcal{I}$ and the rules constructed in (2) (c). Consider an initial tree t shown in Figure 3.8 (b) such that yield$(t) = u_1u_2u_3$. Let t' be a mature derived tree obtained by adjoining $\sigma \in T_n^s(G)$ such that yield$(\sigma) = w_1Bw_2$ to the initial tree t. Then yield$(t') = u_1w_1u_2w_2u_3 = w$ and $w \in L(G)$. On the other hand, the SLMCFG rule for t is $S^{[t]} \rightarrow g[B^{[s]}]$ by construction (2) (c). Remember that there exists a tree $\sigma \in$



(a) (b)

Figure 3.9. Constructed elementary trees

$T_n^s(G)$ for some $n \geq 0$ such that yield$(\sigma) = w_1 B w_2$ if and only if $B^{[s]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$. Thus, $S^{[t]} \stackrel{*}{\Rightarrow}_{G'} g[(w_1, w_2)] = (u_1 w_1 u_2, w_2 u_3)$ and $S_0^{[t]} \stackrel{*}{\Rightarrow}_{G'} J[(u_1 w_1 u_2, w_2 u_3)] = u_1 w_1 u_2 w_2 u_3 = w$. Consequently, $w \in L(G')$ and $L(G) = L(G')$.

$(\mathcal{SLMCFL} \subseteq \mathcal{SLTAL})$ Let $G = (N, T, F, P, S_0)$ be a given SLMCFG. Construct an SLTAG $G' = (N', T, S_0, \mathcal{I}, \mathcal{A})$ as follows:

(1) $N' = N \cup \{X\}$ where $X \notin N$.

(2) $\mathcal{I}$ consists of initial trees shown in Figure 3.9 (a) for $S_0 \rightarrow J[A] \in P$.

(3) $\mathcal{A}$ is the smallest set satisfying:

- For each $A \rightarrow f[B] \in P$ where $f[(x_1, x_2)] = (u_1 x_1 v_1, v_2 x_2 u_2)$, the adjunct tree shown in Figure 3.8 (a) belongs to $\mathcal{A}$.
- For each $A \rightarrow (u_1, u_2) \in P$, the adjunct tree shown in Figure 3.9 (b) belongs to $\mathcal{A}$.

Also, the constraint of every active node is SA($\mathcal{A}$).

Next, we show that $A \stackrel{*}{\Rightarrow}_G (w_1, w_2)$ if and only if there exists a tree $t \in T_n(G')$ for some $n \geq 0$ such that yield$(t) = w_1 A w_2$.

("if" part) By induction on n.

(Basis) If $n = 0$, that is, there exists a mature adjunct tree t such that yield$(t) = w_1 A w_2$, then there should exist a rule $A \rightarrow (w_1, w_2)$ that corresponds to t by construction (3) (see Figure 3.9 (b)). Hence, $A \stackrel{*}{\Rightarrow}_G (w_1, w_2)$ by (L1).

(Induction) Assume that a mature tree $t \in T_n(G)$ is obtained by adjoining a tree

$t_1 \in T_{n-1}(G)$ such that yield$(t_1) = w_1Bw_2$ to an adjunct tree s shown in Figure 3.8 (a). Note that yield$(t) = u_1w_1v_1Av_2w_2u_2$. Then there should exist a rule $A \to f[B]$ where $f[(x_1, x_2)] = (u_1x_1v_1, v_2x_2u_2)$ that corresponds to s by construction (3). By the inductive hypothesis, $B \stackrel{*}{\Rightarrow}_G (w_1, w_2)$. Thus, $A \stackrel{*}{\Rightarrow}_G f[(w_1, w_2)] = (u_1w_1v_1, v_2w_2u_2)$ by (L2).
("only if" part) By induction on the application number of (L1) and (L2).
(Basis) If $A \Rightarrow_G (w_1, w_2)$, i.e., $A \to (w_1, w_2) \in P$, then an adjunct tree t such that yield$(t) = w_1Aw_2$ is constructed by construction (3) (see Figure 3.9 (b)).
(Induction) Assume that $B \stackrel{*}{\Rightarrow}_G (w_1, w_2)$ and $A \stackrel{*}{\Rightarrow}_G f[(w_1, w_2)] = (u_1w_1v_1, v_2w_2u_2)$ by $A \to f[B] \in P$ where $f[(x_1, x_2)] = (u_1x_1v_1, v_2x_2u_2)$. By the inductive hypothesis, there exists a mature derived tree $t_1 \in T_n(G')$ such that yield$(t_1) = w_1Bw_2$. By construction (3), an adjunct tree s shown in Figure 3.8 (a) is in $\mathcal{A}$. Therefore, $s \vdash_{t_1} t$ where yield$(t) = u_1w_1v_1Av_2w_2u_2$.

We can show that $L(G) = L(G')$ in the same way as the proof of ($\mathcal{SLTAL} \subseteq \mathcal{SLMCFL}$). □

3.2 A Subclass of $\mathcal{MCFG}$ for $\mathcal{ESLTAL}$

In this section, we will define a subclass of $(2, 2)$-$\mathcal{MCFG}$ that exactly generates ESLTAL. Let $G = (N, T, S, \mathcal{I}, \mathcal{A})$ be a given ESLTAG. By virtue of Property 2 of [30], we can assume that G is in normal form such that for every semi-simple linear adjunct tree $t \in \mathcal{A}$, yield$(t) \in N$. Thus, for each leaf v of t, either v is the foot node or the label of v is ε (see Figure 3.10). From this observation, we define a subclass of $(2, 2)$-$\mathcal{MCFG}$ by adding rules corresponding to the adjunct trees shown in Figure 3.10 to the definition of SLMCFG.

Definition 3.2. A $(2, 2)$-MCFG $G = (N, T, F, P, S_0)$ is an ESLMCFG if each nonterminating rule has one of the following forms (1) through (3):

(1) $A \to J[B]$ where $\dim(A) = 1$ and $\dim(B) = 2$.

(2) $A \to f[B]$ where f is a simple linear function.

(3) $A \to g[B, D]$ where $\dim(A) = \dim(D) = 2, \dim(B) = 1, g \in \{C_1, C_2, C_3, C_4\}$

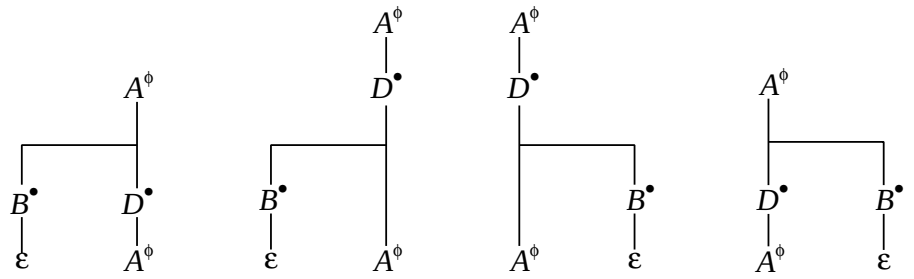


Figure 3.10. Semi-simple linear adjunct trees in normal form

and

$$C_1[x_1,(x_{21},x_{22})] = (x_1x_{21},x_{22}), \quad C_2[x_1,(x_{21},x_{22})] = (x_{21}x_1,x_{22}),$$
$$C_3[x_1,(x_{21},x_{22})] = (x_{21},x_1x_{22}), \quad C_4[x_1,(x_{21},x_{22})] = (x_{21},x_{22}x_1).$$

□

The next lemma establishes the equivalence of $\mathcal{ESLTAL}$ and $\mathcal{ESLMCFL}$. In the proof, we use translations between an ESLTAG and an ESLMCFG similar to those in Lemma 3.1. Let G be a given ESLTAG in normal form. As in Lemma 3.1, initial trees and simple linear adjunct trees in G are translated into ESLMCFG rules with simple linear functions. The semi-simple linear adjunct trees shown in Figure 3.10 (a) through (d) are translated into the ESLMCFG rules of type (3) in Definition 3.2.

Lemma 3.2. $\mathcal{ESLTAL} = \mathcal{ESLMCFL}$.

Proof. ($\mathcal{ESLTAL} \subseteq \mathcal{ESLMCFL}$) Let $G = (N, T, S, \mathcal{I}, \mathcal{A})$ be a given ESLTAG in normal form [30]. We construct an ESLMCFG $G' = (N', T, F, P, S_0)$ from G as follows:

(1) $N' = \{A_1^{[t]}, A_2^{[t]} \mid A \in N,\ t \in \mathcal{I} \cup \mathcal{A}\}$ where $\dim(A_1^{[t]}) = 1$ and $\dim(A_2^{[t]}) = 2$ for $A \in N$ and $t \in \mathcal{I} \cup \mathcal{A}$.

(2) P (and F) are the smallest sets satisfying the following conditions (a) through (d):

(a) For each $A \in N$, $A_1^{[t]} \rightarrow J[A_2^{[t]}] \in P$ for each $t \in \mathcal{I} \cup \mathcal{A}$ and $J[(x_1,x_2)] = x_1x_2 \in F$.

(b) Same as (2) (a) (b) (c) in the proof of ($\mathcal{SLTAL} \subseteq \mathcal{SLMCFL}$) in Lemma 3.1.

(c) For each semi-simple linear adjunct tree t shown in Figure 3.10 (a),

- $A_2^{[t]} \rightarrow C_1[B_1^{[s_1]}, D_2^{[s_2]}] \in P$ for each $s_1 \in \mathcal{T}_1$ and $s_2 \in \mathcal{T}_2$ and $C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22}) \in F$ if B has either SA($\mathcal{T}_1$) or OA($\mathcal{T}_1$) and D has either SA($\mathcal{T}_2$) or OA($\mathcal{T}_2$), and
- $A_2^{[t]} \rightarrow (\varepsilon, \varepsilon) \in P$ if t is mature.

(d) For each semi-simple linear adjunct tree (b) through (d) in Figure 3.10, the rules using C_2, C_3 and C_4, respectively, instead of C_1 belong to P.

Now, we show that there exists a tree $\tau \in T_n^t(G)$ for some $n \geq 0$ such that yield(τ) $= w_1 A w_2$ ($A \in N,\ w_1, w_2 \in T^*$) if and only if $A_2^{[t]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$.
("only if" part) By induction on n.
(Basis) If $\tau = t$ is a mature adjunct tree shown in Figure 3.10, then $A_2^{[t]} \rightarrow (\varepsilon, \varepsilon)$ is constructed by construction (2) (c). Thus, $A_2^{[t]} \stackrel{*}{\Rightarrow}_{G'} (\varepsilon, \varepsilon)$ by (L1). The other cases are the same as the proof of Lemma 3.1.
(Induction) Assume that $\tau \in T_n^t(G) \setminus T_{n-1}^t(G)$ and yield(τ) $= w_1 A w_2$. There are two cases: Either t is a simple linear adjunct tree or t is a semi-simple linear adjunct tree. The proof of the former case is the same as the one in Lemma 3.1. Consider the latter case, then there are four subcases according to the shapes of t as shown in Figure 3.10. We only consider the subcase (a) in Figure 3.10. The other subcases can be treated similarly. Assume that $t \vdash_{\sigma_1} \tau_1 \vdash_{\sigma_2} \tau$ where $\sigma_1 \in T_{n-1}^{s_1}(G),\ \sigma_2 \in T_{n-1}^{s_2}(G)$ and are adjoined at the addresses of B and D respectively. By construction (2) (c), $A_2^{[t]} \rightarrow C_1[B_1^{[s_1]}, D_2^{[s_2]}] \in P$ and by construction (2) (a), $B_1^{[s_1]} \rightarrow J[B_2^{[s_1]}] \in P$. Assume further that yield(σ_1) $= u_1 B u_2$ and yield(σ_2) $= v_1 D v_2$. Then by assumption that yield(τ) $= w_1 A w_2$, we see that $w_1 = u_1 u_2 v_1$ and $w_2 = v_2$. By the inductive hypothesis, $B_2^{[s_1]} \stackrel{*}{\Rightarrow}_{G'} (u_1, u_2)$, $D_2^{[s_2]} \stackrel{*}{\Rightarrow}_{G'} (v_1, v_2)$ and thus $B_1^{[s_1]} \stackrel{*}{\Rightarrow}_{G'} J[(u_1, u_2)] = u_1 u_2$ by (L2). Hence, $A_2^{[t]} \stackrel{*}{\Rightarrow}_{G'} C_1[u_1 u_2, (v_1, v_2)] = (u_1 u_2 v_1, v_2) = (w_1, w_2)$ by (L2).
("if" part) The proof of the "if" part is similar to that of the "only if" part of ($\mathcal{ESLMCFL} \subseteq \mathcal{ESLTAL}$).

We can show that $L(G) = L(G')$ in the same way as the proof of Lemma 3.1.

($\mathcal{ESLMCFL} \subseteq \mathcal{ESLTAL}$) Let $G = (N, T, F, P, S_0)$ be a given ESLMCFG. From G, we construct an ESLTAG $G' = (N \cup \{X\}, T, S_0, \mathcal{I}, \mathcal{A})$ as follows:

(a) (b)

Figure 3.11. Constructed adjunct trees

(1) For each rule $A \rightarrow C_1[B, D] \in P$, add adjunct trees to $\mathcal{A}$ as follows. Note that a rule whose left-hand side is B with $\dim(B) = 1$ has the form of either $B \rightarrow J[E]$ or $B \rightarrow u$ $(u \in T^*)$.

- For each rule $B \rightarrow J[E] \in P$, add the adjunct tree shown in Figure 3.11 (a) to $\mathcal{A}$.
- For each rule $B \rightarrow u \in P$, add the adjunct tree shown in Figure 3.11 (b) to $\mathcal{A}$.

(2) For the rules using C_2, C_3 or C_4, construct adjunct trees in a similar way to (1).

(3) For the other rules, add elementary trees in the same way as (2) and (3) in the proof of $(\mathcal{SLMCFL} \subseteq \mathcal{SLTAL})$ in Lemma 3.1.

We show that $A \stackrel{*}{\Rightarrow}_G (w_1, w_2)$ if and only if there exists a tree $t \in T_n(G')$ for some $n \geq 0$ such that yield$(t) = w_1 A w_2$.
("only if" part) By induction on the application number of (L1) and (L2).
(Basis) If $A \Rightarrow_G (w_1, w_2)$, i.e., $A \rightarrow (w_1, w_2) \in P$, an adjunct tree t such that yield$(t) = w_1 A w_2$ is constructed (see Figure 3.9 (b)).
(Induction) Assume that $E \stackrel{*}{\Rightarrow}_G (u_1, u_2)$, $B \stackrel{*}{\Rightarrow}_G J[(u_1, u_2)] = u_1 u_2$, $D \stackrel{*}{\Rightarrow}_G (v_1, v_2)$ and $A \stackrel{*}{\Rightarrow}_G C_1[u_1 u_2, (v_1, v_2)] = (u_1 u_2 v_1, v_2)$ by $B \rightarrow J[E]$ and $A \rightarrow C_1[B, D]$ in P. By the inductive hypothesis, there exist trees $t_1, t_2 \in T_{n-1}(G')$ such that yield$(t_1) = u_1 E u_2$ and yield$(t_2) = v_1 D v_2$. Since $B \rightarrow J[E]$ and $A \rightarrow C_1[B, D]$ belong to P, the adjunct tree s shown in Figure 3.11 (a) is in $\mathcal{A}$ by construction (1). Thus, we have a tree $t \in T_n(G')$ such that yield$(t) = u_1 u_2 v_1 A v_2$ by adjoining t_1 and t_2 to s. The other cases can be treated in a similar way.

("if" part) The proof of the "if" part is similar to that of the "only if" part of ($\mathcal{ESLTAL} \subseteq \mathcal{ESLMCFL}$).

$L(G) = L(G')$ can be proved in the same way as the proof of Lemma 3.1. □

3.3 A Subclass of $\mathcal{MCFG}$ for $\mathcal{RPL}$

As described in Section 2.2, extra nonterminals in RPGs and functions in MCFGs play a similar role. In this section, we reformulate $\mathcal{RPG}$ as a subclass of $\mathcal{MCFG}$.

Definition 3.3. A $(2,2)$-MCFG $G = (N, T, F, P, S)$ is called an RPG if each nonterminating rule has one of the following forms (1) through (3):

(1) $A \to J[B]$.

(2) $A \to BF[E_1, E_2]$ where $\dim(A) = 2,\ \dim(E_1) = \dim(E_2) = 1$ and $BF[x_1, x_2] = (x_1, x_2)$.

(3) $A \to f[B, D]$ where $\dim(A) = \dim(B) = \dim(D) = 2,\ f \in \{XS_1, XS_2, XS_3, W\}$, XS_i $(i = 1, 2, 3)$ is defined in Example 2.2 and $W[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}x_{21}, x_{22}x_{12})$. □

Note that although an original RPG in Section 2.2 does not have an extra nonterminal corresponding to the functions J and BF, J is used to realize the effect of the homomorphism $h_\wedge$ and BF is used to simulate a production of the form $A \to B \wedge C$.

Proposition 3.3.

$$\mathcal{RPL} \subseteq (2,2)\text{-}\mathcal{MCFL}. \tag{3.6}$$

□

We obtain the following property on recognition complexity.

Proposition 3.4. For a given $w \in T^*$ $(n = |w|)$, whether $w \in L$ or not can be decided in $O(n^6)$ time if L is an RPL, $O(n^5)$ time if L is an ESLTAL, and $O(n^4)$ time if L is an SLTAL.

Proof. For an RPG G, $\deg(G) \leq 6$, for an ESLMCFG G, $\deg(G) \leq 5$ and for an SLMCFG G, $\deg(G) \leq 4$. The proposition follows from Proposition 2.2, Lemmas 3.1 and 3.2. □

The above complexity results were first shown in [30] for $\mathcal{ESLTAL}$ and $\mathcal{SLTAL}$, and in [24] for $\mathcal{RPL}$ by providing an individual recognition algorithm for each class. On the other hand, by identifying these classes of languages as subclasses of $\mathcal{MCFL}$, we can easily obtain the same results as stated in Proposition 3.4. Akutsu [2] defined a structure called a simple pseudoknot and proposed an $O(n^4)$ time exact prediction algorithm and $O(n^{4-\delta})$ time approximation algorithm without using grammar. Note that the set of simple pseudoknots can be generated by an SLTAG.

4. Closure Property

First, we introduce a normal form of ESLMCFG and then show closure properties of $\mathcal{SLTAL}$ and $\mathcal{ESLTAL}$. By using SLMCFG and ESLMCFG, we can prove these properties in a simple way. Some of these properties will be used for proving the inclusion relation between $\mathcal{SLTAL}$ and $\mathcal{ESLTAL}$.

Definition 3.4. An ESLMCFG is in normal form if the following conditions (1) and (2) hold:

(1) For each $A \to f[B]$ where $f[(x_1, x_2)] = (u_1x_1v_1, v_2x_2u_2)$, $|u_1v_1v_2u_2| = 1$.

(2) For each $A \to (u_1, u_2)$ $(u_1, u_2 \in T^*)$, $u_1 = u_2 = \varepsilon$. □

Remark that a similar normal form is defined for ESLTAG in [30]. It is easy to prove the following lemma.

Lemma 3.5. For a given ESLMCFG G, a normal form ESLMCFG G' can be constructed from G such that $L(G) = L(G')$.

Proof. Let $G = (N, T, F, P, S_0)$ be a given ESLMCFG. When we construct a normal form ESLMCFG G' such that $L(G) = L(G')$, we have only to consider that every simple linear function in F can be simulated by functions satisfying the condition (1) in Definition 3.4. For simplicity, we see the following example. Let $A \to f[B] \in P$ where $f[(x_1, x_2)] = (ax_1bc, x_2d) \in F$ $(a, b, c, d \in T)$. This rule can be simulated by $A \to UP^a_{1L}[B_1]$, $B_1 \to UP^c_{1R}[B_2]$, $B_2 \to UP^b_{1R}[B_3]$ and $B_3 \to UP^d_{2R}[B_4]$ where UP^α_{1L}, UP^α_{1R} and UP^α_{2R} $(\alpha \in T)$ are defined in Example 2.2. General case can be treated similarly. Also, every terminating rule in P can be simulated in a similar way. □

Theorem 3.6. $\mathcal{SLTAL}$ and $\mathcal{ESLTAL}$ have the following properties.

(1) $\mathcal{SLTAL}$ contains every linear language.

(2) $\mathcal{SLTAL}$ is closed under union, homomorphism, intersection with regular languages and regular substitution, but is not closed under intersection, concatenation, Kleene closure, positive closure or substitution.

(3) $\mathcal{ESLTAL}$ is closed under intersection with regular languages and substitution, but is not closed under intersection.

Proof. (1) For linear CFG rules $A \to u_1 B v_1$ and $A \to u$, construct SLMCFG rules $A \to f[B]$ where $f[(x_1, x_2)] = (u_1 x_1 v_1, x_2)$ and $A \to (u, \varepsilon)$ respectively.

(2) (union) For two SLMCFGs $G_1 = (N_1, T, F_1, P_1, S_{01})$ and $G_2 = (N_2, T, F_2, P_2, S_{02})$ where $N_1 \cap N_2 = \phi$, let $G' = (N_1 \cup N_2, T, F_1 \cup F_2, P, S_0)$ where P is the union of P_1 and P_2 with S_{01} and S_{02} replaced with S_0. Then $L(G_1) \cup L(G_2) = L(G')$.
(homomorphism) Let $G = (N, T, F, P, S_0)$ be an SLMCFG and h be a homomorphism. An SLMCFG G' such that $h(L(G)) = L(G')$ can be constructed as follows. For a rule $A \to f[B] \in P$ where $f[(x_1, x_2)] = (u_1 x_1 v_1, v_2 x_2 u_2)$, construct a rule $A \to f'[B]$ where $f'[(x_1, x_2)] = (h(u_1) x_1 h(v_1), h(v_2) x_2 h(u_2))$. The construction of terminating rules is similar.
(intersection with regular languages) Same as the proof of Theorem 3.9 (3) of [29].
(regular substitution) Let $G = (N, T, F, P, S_0)$ be an SLMCFG in normal form. We also assume that each rule $A \to f[B] \in P$ has a unique label, say r, and write $r : A \to f[B] \in P$. Let $s : T \to 2^{(T')^*}$ be a regular substitution and for each $\alpha \in T$, let $s(\alpha) = L(G_\alpha)$ where $G_\alpha = (N_\alpha, T', P_\alpha, S_\alpha)$ is a regular grammar. We now construct an SLMCFG $G' = (N', T', F', P', S_0)$ such that $s(L(G)) = L(G')$ as follows. G' will simulate G_α by a linear function instead of generating $\alpha \in T$. To do this, we introduce a nonterminal $X^{[r]}$ in G' where $X \in N_\alpha$ and $r : A \to f[B] \in P$ such that the definition of f contains $\alpha \in T$.

- $N' = N \cup \{X^{[r]} \mid X \in N_\alpha \setminus \{S_\alpha\},\ \alpha \in T,\ r : A \to f[B] \in P\}$.
- F' consists of J, UP_{1L}^{β}, UP_{1R}^{β}, UP_{2L}^{β}, UP_{2R}^{β} $(\beta \in T')$ of Example 2.2 and $EPS[\,] = (\varepsilon, \varepsilon)$.

- P' is the smallest set satisfying:
  - If $S_0 \to J[A] \in P$, $S_0 \to J[A] \in P'$.
  - Assume that $r : A \to f[B] \in P$ where $f[(x_1, x_2)] = (\alpha x_1, x_2)$ $(\alpha \in T)$. If $X \to \beta Y \in P_\alpha$ $(X, Y \in N_\alpha,\ \beta \in T')$, $X^{[r]} \to UP^{\beta}_{1L}[Y^{[r]}] \in P'$, and if $X \to \beta \in P_\alpha$ $(X \in N_\alpha,\ \beta \in T')$, $X^{[r]} \to UP^{\beta}_{1L}[B] \in P'$ where $S^{[r]}_\alpha$ is identified with A for simplicity.
  - For the other rules in P, similar construction can be defined. For example, if $f[(x_1, x_2)] = (x_1, x_2\alpha)$ $(\alpha \in T)$, we will use UP^{β}_{2R} instead of UP^{β}_{1L}.

The proof of $s(L(G)) = L(G')$ is easy. For example, in the second case of the construction of P', $A \overset{*}{\Rightarrow}_G f[(\gamma_1, \gamma_2)] = (\alpha\gamma_1, \gamma_2)$ if and only if $A \overset{*}{\Rightarrow}_{G'} (\xi\gamma_1, \gamma_2)$ for every $\xi \in L(G_\alpha)$.

(intersection) Let $L = \{a_1^n a_2^m a_3^n a_4^n a_5^n \mid m, n \geq 1\}$ and $L' = \{a_1^n a_2^n a_3^n a_4^m a_5^n \mid m, n \geq 1\}$. L and L' belong to $\mathcal{SLTAL}$ (and thus $\mathcal{ESLTAL}$) since each SLMCFG generating L and L', respectively, can be constructed. For example, the SLMCFG generating L is as follows: $S_0 \to J[A]$, $A \to f[A] \mid f[B]$ where $f[(x_1, x_2)] = (a_1x_1a_3, a_4x_2a_5)$ and $B \to g[B] \mid (a_2, \varepsilon)$ where $g[(x_1, x_2)] = (a_2x_1, x_2)$. The SLMCFG generating L' is constructed in a similar way. The intersection of them, i.e., $L \cap L' = \{a_1^n a_2^n a_3^n a_4^n a_5^n \mid n \geq 1\}$ is not a 2-MCFL by Lemma 3.3 of [29]. Therefore, L does not belong to $\mathcal{SLTAL}$.

(concatenation) Let $L = \{\sharp a_1^k b_1^k \sharp a_2^l b_2^l \mid k, l \geq 1\}$ and $L' = \{\sharp a_3^m b_3^m \sharp a_4^n b_4^n \sharp \mid m, n \geq 1\}$, both of which are SLTALs. An SLMCFG generating L is such that $S_0 \to J[S]$, $S \to \text{add}^{\sharp\sharp}[A]$ where $\text{add}^{\sharp\sharp}[(x_1, x_2)] = (\sharp x_1, \sharp x_2)$, $A \to f[A] \mid B$ where $f[(x_1, x_2)] = (a_1x_1b_1, x_2)$ and $B \to g[B] \mid (a_1b_1, a_2b_2)$ where $g[(x_1, x_2)] = (x_1, a_2x_2b_2)$. The construction of an SLMCFG generating L' is similar. The concatenation of them, i.e., $LL' = L_2$ defined in (3.3) is not an SLTAL.

(Kleene closure, positive closure) By the next corollary, $\mathcal{SLTAL}$ is a union closed full trio. If $\mathcal{SLTAL}$ is closed under Kleene closure or positive closure, then $\mathcal{SLTAL}$ is closed under concatenation by Proposition 3.8, which is a contradiction.

(substitution) Let $L_1 = \{\sharp d_1 \sharp d_2 \sharp d_3 \sharp d_4 \sharp\}$, which is a finite language and thus an

SLTAL, and let s be a substitution such that $s(d_i) = \{a_i^n b_i^n \mid n \geq 1\}$ $(1 \leq i \leq 4)$, which is also an SLTAL by (1) of this theorem. Then $s(L_1) = L_2$ defined in (3.3), which is not an SLTAL.

(3) (intersection with regular languages) Same as the proof of Theorem 3.9 (3) of [29].
(substitution) Let $G = (N, T, F, P, S_0)$ be an ESLMCFG in normal form. Let s be a substitution such that $s(a) = L(G_a)$ $(a \in T)$ where $G_a = (N_a, T', F_a, P_a, S_{0a})$ is an ESLMCFG without sharing nonterminals with one another and with G. An ESLMCFG $G' = (N \cup \bigcup_{a \in T} N_a \cup \{X\}, T', F \cup \bigcup_{a \in T} F_a, P' \cup \bigcup_{a \in T} P_a, S_0)$ generates $s(L(G))$ where P' is the same as P except that for a rule $A \to f[B]$ where $f[(x_1, x_2)] = (ax_1, x_2)$, P' contains $A \to C_1[S_{0a}, B]$ instead of $A \to f[B]$, and similarly for the other rules that use simple linear functions.
(intersection) See (2) of this proof. □

A class of languages is called a *full trio* (or *cone*) if it is closed under homomorphism, inverse homomorphism and intersection with regular languages. A full trio closed under union, concatenation and Kleene closure is called a *full abstract family of languages* (*full* AFL). When we try to show that a class of languages is a full trio (or full AFL), major difficulty lies in showing closure under inverse homomorphism. The following propositions [19] present an alternative way of proving it.

Proposition 3.7 ([19])**.** If a class of languages is closed under ε-free regular substitution, linear erasing, union with regular languages and intersection with regular languages, then it is closed under inverse homomorphism. The same conclusion can be made for ε-free classes even without assuming closure under union with regular languages. □

Proposition 3.8 ([19])**.** If a class of languages includes a language containing a nonempty word and is closed under union, Kleene closure (or positive closure), ε-free regular substitution, intersection with regular languages and homomorphism, then the class is a full AFL. If a class of languages contains all regular languages and is closed under substitution as well as under intersection with regular languages, then the class is a full AFL. □

Corollary 3.9. $\mathcal{SLTAL}$ is a full trio. $\mathcal{ESLTAL}$ is a substitution closed full AFL.

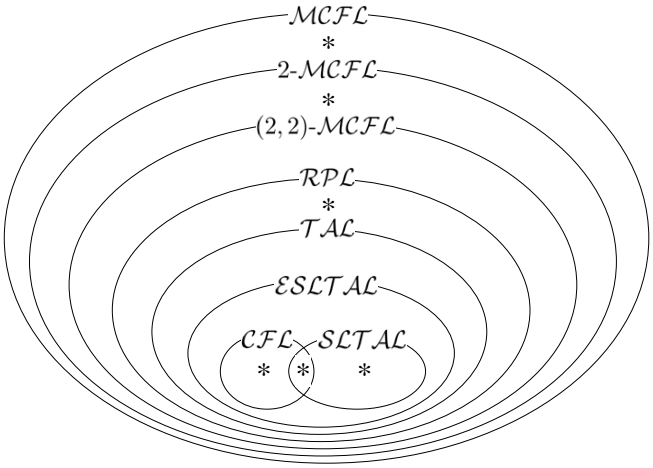


Figure 3.12. Known results on inclusion relation

Proof. (full trio) By Proposition 3.7 and Theorem 3.6 (2) .
(full AFL) By Proposition 3.8 and Theorem 3.6 (1), (3). □

5. Inclusion Relation

First, we summarize the inclusion relations between the classes of languages stated in (3.1) through (3.6) (see Figure 3.12). In the figure, an asterisk indicates that there exists at least one language in the region where the asterisk is placed.

Proposition 3.10. (1) $(\mathcal{CFL} \cup \mathcal{SLTAL}) \subseteq \mathcal{ESLTAL} \subseteq \mathcal{TAL} \subsetneq (2,2)\text{-}\mathcal{MCFL}$.

(2) $\mathcal{RPL} \subseteq (2,2)\text{-}\mathcal{MCFL} \subsetneq 2\text{-}\mathcal{MCFL} \subsetneq \mathcal{MCFL}$. □

In the following, we refine the above proposition.

5.1 $\mathcal{RPL} = (2,2)$-$\mathcal{MCFL}$

We introduce the following condition (S) that states that for each argument (x_{i1}, x_{i2}) of a function of a 2-MCFG, the order of the occurrences of its components x_{i1} and x_{i2} is not interchanged in the function value.

(S) Let $G = (N, T, F, P, S)$ be a 2-MCFG and $f : (T^*)^{d_1} \times \cdots \times (T^*)^{d_k} \to (T^*)^{d_0}$ $(1 \leq d_i \leq 2$ for $0 \leq i \leq k)$ be an arbitrary function in F defined by (see (F) in Section

2 of Chapter 2):

$$f^{[h]}[z_1, \ldots, z_k] = \alpha_h \ (1 \leq h \leq d_0)$$

where either $z_i = x_i$ or $z_i = (x_{i1}, x_{i2})$ $(1 \leq i \leq k)$. Let $\varphi = \alpha_1$ if $d_0 = 1$ and let $\varphi = \alpha_1\alpha_2$ if $d_0 = 2$. For each i $(1 \leq i \leq k)$, if $z_i = (x_{i1}, x_{i2})$ and both x_{i1} and x_{i2} occur in φ, then x_{i1} occurs to the left of the occurrence of x_{i2}, i.e., $\varphi = \xi_1 x_{i1} \xi_2 x_{i2} \xi_3$ for some ξ_j $(1 \leq j \leq 3)$.

Lemma 3.11. For a given 2-MCFG G, we can construct a 2-MCFG G' satisfying the condition (S) and $L(G) = L(G')$.

Proof. Let $G = (N, T, F, P, S)$ be a given 2-MCFG. For example, if there exits a rule $A \to f[B, D] \in P$ where $f[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{12}x_{21}, x_{22}x_{11})$, then the pair of variables (x_{11}, x_{12}) violates the condition (S). We interchange the occurrences of x_{11} and x_{12} in the definition of f to obtain another function, say f_s such that $f_s[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}x_{21}, x_{22}x_{12})$. Then $A \to f[B, D]$ is replaced with $A \to f_s[B^R, D]$ where B^R is a new nonterminal symbol such that $B \overset{*}{\Rightarrow} (u_1, u_2)$ if and only if $B^R \overset{*}{\Rightarrow} (u_2, u_1)$. For B^R to satisfy this property, more rules should be added.

Generally, we will construct a 2-MCFG $G' = (N', T, F', P', S)$ as

$$N' = N \cup \{A^R \mid A \in N\},$$
$$F' = \{f_s, f_s^R \mid f \in F\},$$
$$P' = \{A_0 \to f_s[B_1, \ldots, B_k],\ A_0^R \to f_s^R[D_1, \ldots, D_k] \mid A_0 \to f[A_1, \ldots, A_k] \in P\}$$

where f_s, f_s^R, B_i and D_i $(1 \leq i \leq k)$ are defined as follows. Let the definition of f be the one stated in the condition (S). Also, let $\varphi^R = \varphi = \alpha_1$ if $d_0 = 1$ and let $\varphi^R = \alpha_2\alpha_1$ if $d_0 = 2$.

$$B_i = \begin{cases} A_i^R & \text{if } x_{i2} \text{ occurs to the left of } x_{i1} \text{ in } \varphi \\ A_i & \text{otherwise.} \end{cases}$$

$$D_i = \begin{cases} A_i^R & \text{if } x_{i2} \text{ occurs to the left of } x_{i1} \text{ in } \varphi^R \\ A_i & \text{otherwise.} \end{cases}$$

Assume that $d_0 = 2$, i.e., $f[z_1, \ldots, z_k] = (\alpha_1, \alpha_2)$. (The case that $d_0 = 1$ can be treated in the same way.) Then $f_s[z_1, \ldots, z_k] = (\beta_1, \beta_2)$ and $f_s^R[z_1, \ldots, z_k] = (\gamma_2, \gamma_1)$. Here, β_j $(j = 1, 2)$ is obtained from α_j by interchanging the occurrences of x_{i1} and x_{i2} $(1 \leq i \leq k)$ if and only if x_{i2} occurs to the left of x_{i1} in φ. Similarly, γ_j $(j = 1, 2)$

is obtained from α_j by interchanging the occurrences of x_{i1} and x_{i2} $(1 \leq i \leq k)$ if and only if x_{i2} occurs to the left of x_{i1} in φ^R.

Now, we show by induction on the application number of (L1) and (L2) that $A \stackrel{*}{\Rightarrow}_G (u_1, u_2)$ $(u_1, u_2 \in T^*)$ if and only if $A \stackrel{*}{\Rightarrow}_{G'} (u_1, u_2)$ and $A^R \stackrel{*}{\Rightarrow}_{G'} (u_2, u_1)$.
("only if" part) (Basis) If $A \Rightarrow_G (u_1, u_2)$, $A \to (u_1, u_2) \in P$. By the above construction of G', $A \to (u_1, u_2) \in P'$ and $A^R \to (u_2, u_1) \in P'$. Hence, $A \Rightarrow_{G'} (u_1, u_2)$ and $A^R \Rightarrow_{G'} (u_2, u_1)$.
(Induction) Assume that $A_i \stackrel{*}{\Rightarrow}_G (v_{i1}, v_{i2})$ $(1 \leq i \leq k)$ and $A_0 \stackrel{*}{\Rightarrow}_G f[(v_{11}, v_{12}), \ldots, (v_{k1}, v_{k2})] = (u_1, u_2)$ by $A_0 \to f[A_1, \ldots, A_k] \in P$. By the above construction, $A_0 \to f[A_1, \ldots, A_k] \in P$ is replaced with $A_0 \to f_s[B_1, \ldots, B_k]$ and $A_0^R \to f_s^R[D_1, \ldots, D_k]$ in P'. If x_{i2} occurs to the left of x_{i1} in φ, then $B_i = A_i^R$, otherwise $B_i = A_i$. By the inductive hypothesis, $A_i \stackrel{*}{\Rightarrow}_{G'} (v_{i1}, v_{i2})$ and $A_i^R \stackrel{*}{\Rightarrow}_{G'} (v_{i2}, v_{i1})$. Thus, $A_0 \stackrel{*}{\Rightarrow}_{G'} (u_1, u_2)$. $A_0^R \stackrel{*}{\Rightarrow}_{G'} (u_2, u_1)$ can be shown in the same way.
("if" part) The proof of the "if" part is similar.

$L(G) = L(G')$ can be proved by considering a derivation from the start symbol. □

Lemma 3.12. Let $G = (N, T, F, P, S)$ be a $(2, 2)$-MCFG satisfying the condition (S). Then we can construct an RPG G' such that $L(G) = L(G')$.

Proof. Let $G = (N, T, F, P, S)$ be an arbitrary $(2, 2)$-MCFG satisfying the condition (S). We construct an RPG G' weakly equivalent to G as follows. The number of functions $f : (T^*)^2 \times (T^*)^2 \to (T^*)^2$ satisfying the condition (S) is 18. A half of them can be obtained from the other half of them by interchanging the first and the second arguments. Among the remaining nine functions, four are RPG functions. The others are:

$$f_1[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}, x_{12}x_{21}x_{22}),$$
$$f_2[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}x_{12}, x_{21}x_{22}),$$
$$f_3[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}x_{12}x_{21}, x_{22}),$$
$$f_4[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}, x_{21}x_{22}x_{12}),$$
$$f_5[(x_{11}, x_{12}), (x_{21}, x_{22})] = (x_{11}x_{21}x_{22}, x_{12}).$$

These functions can be simulated by RPG functions (see Table 3.1). Also, a simpler function such as $g : (T^*)^2 \times T^* \to (T^*)^2$ can be treated in the same way. □

By Proposition 3.10 (2), Lemmas 3.11 and 3.12, we obtain the following theorem.

Table 3.1. Simulation of $(2,2)$-MCFG functions by RPG functions

$(2,2)$-MCFG rule	Corresponding RPG rule
$A \to f_1[B,D]$	$A \to XS_2[B,Y_1],\ Y_1 \to BF[Y_2,Y_3],\ Y_2 \to \varepsilon,\ Y_3 \to J[D]$
$A \to f_2[B,D]$	$A \to BF[Y_1,Y_2],\ Y_1 \to J[B],\ Y_2 \to J[D]$
$A \to f_3[B,D]$	$A \to XS_2[Y_1,D],\ Y_1 \to BF[Y_2,Y_3],\ Y_2 \to J[B],\ Y_3 \to \varepsilon$
$A \to f_4[B,D]$	$A \to W[B,Y_1],\ Y_1 \to BF[Y_2,Y_3],\ Y_2 \to \varepsilon,\ Y_3 \to J[D]$
$A \to f_5[B,D]$	$A \to W[B,Y_1],\ Y_1 \to BF[Y_2,Y_3],\ Y_2 \to J[D],\ Y_3 \to \varepsilon$

Theorem 3.13. $\mathcal{RPL} = (2,2)$-$\mathcal{MCFL}$. □

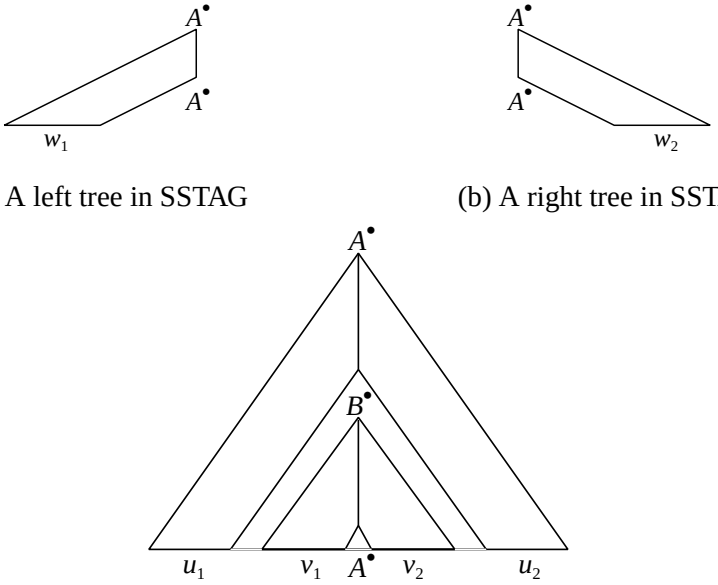
5.2 $\mathcal{ESLTAL} = \mathcal{SSTAL} = (2,2)$-$\mathcal{MCFL}$ with degree ≤ 5

Next, we will show the equivalence of $\mathcal{ESLTAL}$, $\mathcal{SSTAL}$ and $(2,2)$-$\mathcal{MCFL}$ whose degree is five or less.

Theorem 3.14. $\mathcal{ESLTAL} = \mathcal{SSTAL} = (2,2)$-$\mathcal{MCFL}$ with degree ≤ 5.

Proof. $(\mathcal{ESLTAL} \subseteq \mathcal{SSTAL})$ To prove this inclusion relation, we redefine SSTAG in a different way from the original mentioned in Section 2.1. Let $G = (N, T, S, \mathcal{I}, \mathcal{A})$ be a TAG. We divide $\mathcal{A}$ into three finite sets $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ arbitrarily and call each element of them a left tree, a right tree and a wrapping tree respectively. If each left tree satisfies (LT1) and (LT2), each right tree satisfies (RT1) and (RT2), and each wrapping tree satisfies (WT2), then G is called an SSTAG. Note that $\mathcal{SSTAG}$ defined in [28] is included in the class of grammars we redefined above, but not vice versa. Part of the reason is because in the original definition of SSTAG, every adjunct tree satisfying (LT1) is called a left tree and every left tree must satisfy (LT2), while in our definition, we can classify every adjunct tree satisfying (LT1) into a wrapping tree. Here, we classify every adjunct tree in a given ESLTAG into a wrapping tree and assume that the other sets $\mathcal{A}_1$ and $\mathcal{A}_2$ are empty. Then by the definition of ESLTAG, every wrapping tree in $\mathcal{A}_3$ satisfies (WT2) and thus G is an SSTAG.

$((2,2)$-$\mathcal{MCFL}$ with degree $\leq 5 \subseteq \mathcal{ESLTAL})$ By virtue of Lemmas 3.2 and 3.11, we have only to consider translation from a $(2,2)$-MCFG G such that $\deg(G) \leq 5$ and G satisfies the condition (S) into an ESLMCFG G'. The construction of G' from G is as follows. The number of functions satisfying the condition (S) is 12. A half of them



(a) A left tree in SSTAG

(b) A right tree in SSTAG

(c) A wrapping tree in SSTAG

Figure 3.13. Adjunct trees in SSTAG

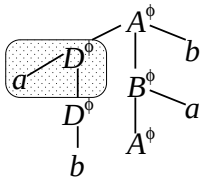


Figure 3.14. A mature derived tree in SSTAG

can be obtained from the other half of them by interchanging the first and the second arguments. Among the remaining six functions, four are ESLMCFG ones. The others are $g_1[x_1,(x_{21},x_{22})] = (x_1, x_{21}x_{22})$ and $g_2[x_1,(x_{21},x_{22})] = (x_{21}x_{22}, x_1)$. The rule $A \rightarrow g_1[B,D]$ in G can be simulated by $A \rightarrow C_1[B,Y_1]$, $Y_1 \rightarrow C_4[Y_2,Y_3]$, $Y_2 \rightarrow J[D]$ and $Y_3 \rightarrow (\varepsilon,\varepsilon)$ in G'. Similarly, $A \rightarrow g_2[B,D]$ can be simulated by $A \rightarrow C_4[B,Y_1]$, $Y_1 \rightarrow C_1[Y_2,Y_3]$, $Y_2 \rightarrow J[D]$ and $Y_3 \rightarrow (\varepsilon,\varepsilon)$.

($\mathcal{SSTAL} \subseteq (2,2)$-$\mathcal{MCFL}$ with degree ≤ 5) Let $G = (N,T,S,\mathcal{I},\mathcal{A})$ be a given SSTAG defined in [28]. We construct a $(2,2)$-MCFG $G' = (N',T,F,P,S_0)$ with $\deg(G') \leq 5$ as follows:

(1) $N' = \{A_1, A_2, A_1^{[t]}, A_2^{[t]} \mid A \in N,\ t \in \mathcal{I} \cup \mathcal{A}\}$ where $\dim(A_1) = \dim(A_1^{[t]}) = 1$

and $\dim(A_2) = \dim(A_2^{[t]}) = 2$. In addition, we classify $B_1 \in N'$ corresponding to the wrapping node of a wrapping tree into four nonterminals B_{LU}, B_{LD}, B_{RU} and B_{RD}. Their dimensions are one and each of them derives a portion of the tree yield, say, $B_{LU} \overset{*}{\Rightarrow}_{G'} u_1$, $B_{LD} \overset{*}{\Rightarrow}_{G'} v_1$, $B_{RU} \overset{*}{\Rightarrow}_{G'} u_2$ and $B_{RD} \overset{*}{\Rightarrow}_{G'} v_2$ where $u_1, u_2, v_1, v_2 \in T^*$ (see Figure 3.13 (c)).

(2) P (and F) are the smallest sets satisfying the following conditions:

(a) $S_0 \to J[S_2^{[t]}] \in P$ for each $t \in \mathcal{I}$ and $J[(x_1, x_2)] = x_1 x_2 \in F$.

(b) For each $A \in N$, $A_1 \to J[A_2], A_1^{[t]} \to J[A_2^{[t]}] \in P$ for each $t \in \mathcal{I} \cup \mathcal{A}$ and $J \in F$.

(c) For each left tree $t \in \mathcal{A}$ shown in Figure 3.13 (a),

- $A_2^{[t]} \to C_1[A_1^{[s]}, A_2^{[t]}] \in P$ for each left tree $s \in \mathcal{T}$ and $C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22}) \in F$ if the root node has either SA($\mathcal{T}$) or OA($\mathcal{T}$),
- $A_2^{[t]} \to C_2[A_1^{[s]}, A_2^{[t]}] \in P$ for each left tree $s \in \mathcal{T}$ and $C_2[x_1, (x_{21}, x_{22})] = (x_{21} x_1, x_{22}) \in F$ if the foot node has either SA($\mathcal{T}$) or OA($\mathcal{T}$),
- $A_2^{[t]} \to C_4[A_1^{[s]}, A_2^{[t]}] \in P$ for each right tree $s \in \mathcal{A}$ and $C_4[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22} x_1) \in F$ if s is adjoined to the root node of t,
- $A_2^{[t]} \to C_3[A_1^{[s]}, A_2^{[t]}] \in P$ for each right tree $s \in \mathcal{A}$ and $C_3[x_1, (x_{21}, x_{22})] = (x_{21}, x_1 x_{22}) \in F$ if s is adjoined to the foot node of t,
- See (e) if we adjoin an adjunct tree to a node that is not in the backbone, and
- $A_2^{[t]} \to (w_1, \varepsilon)$ if t is mature.

(d) For each right tree $t \in \mathcal{A}$ shown in Figure 3.13 (b),

- $A_2^{[t]} \to C_4[A_1^{[s]}, A_2^{[t]}] \in P$ for each right tree $s \in \mathcal{T}$ and $C_4[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22} x_1) \in F$ if the root node has either SA($\mathcal{T}$) or OA($\mathcal{T}$),
- $A_2^{[t]} \to C_3[A_1^{[s]}, A_2^{[t]}] \in P$ for each right tree $s \in \mathcal{T}$ and $C_3[x_1, (x_{21}, x_{22})] = (x_{21}, x_1 x_{22}) \in F$ if the foot node has either SA($\mathcal{T}$) or OA($\mathcal{T}$),
- $A_2^{[t]} \to C_1[A_1^{[s]}, A_2^{[t]}] \in P$ for each left tree $s \in \mathcal{A}$ and $C_1[x_1, (x_{21}, x_{22})] = (x_1 x_{21}, x_{22}) \in F$ if s is adjoined to the root node of t,
- $A_2^{[t]} \to C_2[A_1^{[s]}, A_2^{[t]}] \in P$ for each left tree $s \in \mathcal{A}$ and $C_2[x_1, (x_{21}, x_{22})] = (x_{21} x_1, x_{22}) \in F$ if s is adjoined to the foot node of t,

- See (e) if we adjoin an adjunct tree to a node that is not in the backbone, and
- $A_2^{[t]} \rightarrow (\varepsilon, w_2)$ if t is mature.

(e) For each wrapping tree $t \in \mathcal{A}$,

- $A_2^{[t]} \rightarrow C_1[B_{LU}, X_1], X_1 \rightarrow C_4[B_{RU}, X_2], X_2 \rightarrow C_3[B_{RD}, X_3], X_3 \rightarrow C_2[B_{LD}, B_2^{[s]}] \in P$ for each wrapping tree $s \in \mathcal{T}$ and $C_1, C_2, C_3, C_4 \in F$ if the wrapping node B in Figure 3.13 (c) has either SA($\mathcal{T}$) or OA($\mathcal{T}$),
- $A_2^{[t]} \rightarrow C_1[D_1, A_2], D_2 \rightarrow (a, \varepsilon), A_2 \rightarrow (b, ab) \in P$ where $a, b \in T$ and $C_1 \in F$ if we adjoin the adjunct tree indicated by the shaded region in Figure 3.14 to D that is not in the backbone and then the derived tree is mature. The other examples and the adjunction of a left tree or a right tree to a node in the backbone of t can be treated similarly, and
- $A_2^{[t]} \rightarrow (w_1, w_2)$ if t is mature.

(f) For each initial tree $t \in \mathcal{I}$, adjoin derived trees to t and then construct rules for the mature tree in a similar way to (e).

Although the derivation in SSTAG is not specifically described in [28], the recognition algorithm is correct and thus we can construct G' by the above construction.

Next, we show that there exists a tree $\tau \in T_n^t(G)$ for some $n \geq 0$ such that yield$(\tau) = w_1 A w_2$ $(A \in N,\ w_1, w_2 \in T^*)$ if and only if $A_2^{[t]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$.
("only if" part) By induction on n.
(Basis) If $\tau = t$ is a mature wrapping tree shown in Figure 3.13 (c), then a rule $A_2^{[t]} \rightarrow (w_1, w_2)$ is constructed by construction (2) (e). Thus, $A_2^{[t]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$ by (L1). The other cases can be proved in the same way.
(Induction) Assume that $\tau \in T_n^t(G) \setminus T_{n-1}^t(G)$. We consider only the case where t is a wrapping tree and the adjunction is performed only at the wrapping node (see Figure 3.13 (c)). The other cases can be treated similarly. Assume that $t \vdash_\sigma \tau$ where $\sigma \in T_{n-1}^s(G)$ is adjoined to the wrapping node B in t and yield$(\sigma) = w_1 B w_2$. (Note that s is a wrapping tree.) Then yield$(\tau) = u_1 w_1 v_1 A v_2 w_2 u_2$. By construction (2) (e), $A_2^{[t]} \rightarrow C_1[B_{LU}, X_1], X_1 \rightarrow C_4[B_{RU}, X_2], X_2 \rightarrow C_3[B_{RD}, X_3], X_3 \rightarrow C_2[B_{LD}, B_2^{[s]}] \in P$. By the inductive hypothesis, $B_2^{[s]} \stackrel{*}{\Rightarrow}_{G'} (w_1, w_2)$. Then $X_3 \stackrel{*}{\Rightarrow}_{G'}$

$C_2[v_1,(w_1,w_2)] = (w_1v_1,w_2)$, $X_2 \overset{*}{\Rightarrow}_{G'} C_3[v_2,(w_1v_1,w_2)] = (w_1v_1,v_2w_2)$, $X_1 \overset{*}{\Rightarrow}_{G'} C_4[u_2,(w_1v_1,v_2w_2)] = (w_1v_1,v_2w_2u_2)$ and thus $A_2^{[t]} \overset{*}{\Rightarrow}_{G'} C_1[u_1,(w_1v_1,v_2w_2u_2)] = (u_1w_1v_1,v_2w_2u_2)$ by (L2).

("if" part) The proof of the "if" part is similar.

$L(G) = L(G')$ can be shown by considering the correspondence between the initial trees in $\mathcal{I}$ and the rules constructed in (2) (f). □

5.3 $(\mathcal{CFL} \cup \mathcal{SLTAL}) \subsetneq \mathcal{ESLTAL}$

We show the inclusion relation between $\mathcal{SLTAL}$ and $\mathcal{ESLTAL}$.

Theorem 3.15. Let $L_3 = \{\sharp a_1^k b_1^k c_1^k \sharp a_2^l b_2^l c_2^l \sharp a_3^m b_3^m c_3^m \sharp a_4^n b_4^n c_4^n \sharp \mid k,l,m,n \geq 1\}$. Then $L_3 \in \mathcal{ESLTAL} \setminus (\mathcal{CFL} \cup \mathcal{SLTAL})$.

Proof. Let h_1 be a homomorphism such that $h_1(a_1) = a_1$, $h_1(b_1) = b_1$, $h_1(c_1) = c_1$ and $h_1(x) = \varepsilon$ for $x \in \{a_i,b_i,c_i \mid i = 2,3,4\} \cup \{\sharp\}$. Then $h_1(L_3) = \{a_1^k b_1^k c_1^k \mid k \geq 1\}$, which is not a CFL. Since $\mathcal{CFL}$ is closed under homomorphism, L_3 is not a CFL. Similarly, let h_2 be a homomorphism such that $h_2(c_i) = \varepsilon$ for $i = 1,2,3$ and identity on the other symbols. Then $h_2(L_3) = L_2$ defined in (3.3), which is not an SLTAL. Since $\mathcal{SLTAL}$ is closed under homomorphism by Theorem 3.6 (2), L_3 is not an SLTAL. Next, we give an ESLMCFG (with start symbol S_0) generating L_3 as follows:

$S_0 \to J[S_1]$, $J[(x_1,x_2)] = x_1x_2$,
$S_1 \to \text{add}^\sharp[T_1]$, $\text{add}^\sharp[(x_1,x_2)] = (\sharp x_1, x_2)$,
$T_i \to C_4[S_{i+1},A_i]\ (1 \leq i \leq 3)$, $C_4[x_1,(x_{21},x_{22})] = (x_{21},x_{22}x_1)$,
$S_2 \to J[T_2]$,
$S_3 \to J[T_3]$,
$S_4 \to J[A_4]$,
$A_i \to f_i[A_i] \mid (a_ib_i, c_i\sharp)\ (1 \leq i \leq 4)$, $f_i[(x_1,x_2)] = (a_ix_1b_i, c_ix_2)\ (1 \leq i \leq 4)$. □

Finally, we sum up the inclusion relations obtained in this section. The following corollary follows from Proposition 3.10, Theorems 3.13, 3.14 and 3.15 (see Figure 3.15).

Corollary 3.16. $(\mathcal{CFL} \cup \mathcal{SLTAL}) \subsetneq \mathcal{ESLTAL} = \mathcal{SSTAL} = (2,2)\text{-}\mathcal{MCFL}$ with degree $\leq 5 \subseteq \mathcal{TAL} \subsetneq \mathcal{RPL} = (2,2)\text{-}\mathcal{MCFL} \subsetneq 2\text{-}\mathcal{MCFL} \subsetneq \mathcal{MCFL}$. □

Whether the inclusion $\mathcal{ESLTAL} \subseteq \mathcal{TAL}$ is proper or not is an open problem.

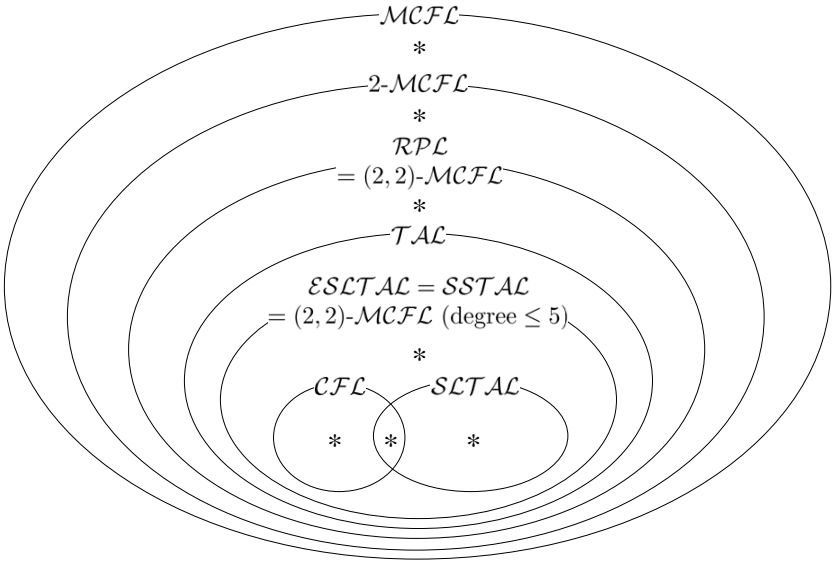


Figure 3.15. New results on inclusion relation

6. Summary

In this chapter, several formal grammars for describing RNA secondary structure with pseudoknots were identified as subclasses of $\mathcal{MCFG}$, and their generative powers were compared. To the author's knowledge, the exact definition of RNA pseudoknot in a biological or geometrical sense is not known, and then it is difficult to answer which class of grammars is the minimum to represent pseudoknots. However, SLTAGs cannot generate RNA sequences obtained by repeating a simple pseudoknot shown in Figure 1.2 (b) by (3.3), and by virtue of Theorem 3.15, ESLTAGs (equivalently, ESLMCFGs or $(2,2)$-MCFGs with degree 5 or less) are candidates for the minimum grammars that can represent repeating simple pseudoknots. We also showed that $\mathcal{SLTAL}$ is a full trio and $\mathcal{ESLTAL}$ is a substitution closed full AFL, which is a good property from the formal language theoretical point of view.

As described in the previous section, we conjecture that $\mathcal{TAL}$ properly includes $\mathcal{ESLTAL}$. To show this, it is sufficient to find a language that belongs to $\mathcal{TAL}$ but not to $\mathcal{ESLTAL}$ by using a pumping lemma for $\mathcal{ESLTAL}$. As has been mentioned in Section 1, the final goal of this study is to predict RNA secondary structure including pseudoknots by using an appropriate subclass of $\mathcal{MCFG}$. For that purpose, we must introduce probabilistic models and design algorithms for them. For instance, stochastic

CFGs (SCFGs) where probability is assigned to each rule are used for RNA secondary structure prediction without pseudoknots [8, 9, 26]. To apply subclasses of $\mathcal{MCFG}$ to RNA secondary structure prediction with pseudoknots, probabilistic extension of MCFGs should be introduced like SCFGs, which is presented in Chapter 4. It is especially important to consider the way to give probabilities to MCFG rules in order to obtain biologically realistic structure. In addition, we would like to consider using information on known secondary structures from some databases.

Chapter 4

Analysis of RNA Pseudoknotted Structure Using SMCFGs

1. Introduction

Recently, it has been thought that most of the RNAs transcribed from genome sequences are non-coding RNAs (ncRNAs), and much attention has been paid to their structures and functions. Non-coding RNAs fold into characteristic structures in such a way that canonical Watson-Crick base pairs and non-canonical pairs bond each other. The resulting base paired structure is called the secondary structure. In typical RNA secondary structures, base pairs occur in a nested way, that is, for all positions (i, j) and (i', j') indicating base pairs in one stem, either $i < i' < j' < j$ or $i' < i < j < j'$ holds. On the other hand, there are substructures where some base pairs occur in a crossed fashion, which are called *pseudoknots*, and they are found in several RNAs such as rRNAs, tmRNAs and viral RNAs. 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. Another is to take

free energy minimization into account. Eddy and Durbin [9], and Sakakibara et al. [26] 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 dependency of base pairs in pseudoknots (Figure 1.2 (b)) due to the lack of generative power. Brown and Wilson [5] proposed a model based on intersections of SCFGs to describe RNA pseudoknots. Cai et al. [6] introduced a model based on parallel communication grammar systems using a single CFG synchronized with a number of regular grammars. Akutsu [2] 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 [23, 24] 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 gaps. Uemura et al. [30] defined specific subclasses of tree adjoining grammars (TAGs) named simple linear TAGs (SLTAGs) and extended SLTAGs (ESLTAGs), respectively, and predicted RNA pseudoknots by using the parsing algorithm of ESLTAGs. Matsui et al. [20] 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. For another application using a grammar-based model, Rivas and Eddy [25] presented the detection of ncRNA genes in genome sequences. They tested the maximum liklihood scanning algorithm based on SCFGs for some genome sequences including known RNA genes such as tRNAs.

In Chapter 3, we have identified RPGs, SLTAGs and ESLTAGs as subclasses of *multiple context-free grammars* (MCFGs) [14, 29], and have shown a candidate subclass for the minimum grammars to represent pseudoknots. The remainder of this chapter is organized as follows. First, stochastic MCFGs (SMCFGs) are introduced in Section 2, which are probabilistic extension of MCFGs. In Section 3, we present a polynomial time parsing algorithm for finding the most probable derivation tree and a probability parameter estimation algorithm based on the EM algorithm. In Section 4, we show some experimental results on pseudoknot prediction for three viral RNA

families using the SMCFG parsing algorithm. Furthermore, we perform ncRNA gene finding for several genome sequences known to have ncRNA genes with pseudoknots. Experimental results are discussed in Section 5. Section 6 concludes this chapter.

2. Stochastic Multiple Context-Free Grammar

Stochastic multiple context-free grammars (stochastic MCFGs, or SMCFGs) are probabilistic extensions of MCFGs. An SMCFG is a 5-tuple $G = (N, T, F, P, S)$ where the definitions of N, T, F and S are the same as those of MCFG. P is a finite set of (production) rules associated with some real numbers and each rule in P has the form of $A_0 \xrightarrow{p} f[A_1, \ldots, A_k]$ where $A_i \in N$ $(0 \leq i \leq k)$, $f \in F$ and p is a real number with $0 < p \leq 1$ 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 $A_0 \rightarrow f[A_1, \ldots, A_k]$.

We next define derivation trees of SMCFGs 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 that 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) that 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 $(0 < q \leq 1)$, we write $A \stackrel{*}{\Rightarrow} \overline{\alpha}$ 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 $L(G) = \{w \in T^* \mid S \stackrel{*}{\Rightarrow} w$ with probability greater than $0\}$. Dimension, rank and degree of SMCFG are defined in the same way as those of MCFG.

Example 4.1. Let $G_1 = (N_1, T_1, F_1, P_1, S)$ be a $(2, 1)$-SMCFG where $N_1 = \{S, A\}$, $T_1 = \{a, b\}$, $P_1 = \{S \xrightarrow{1} J[A],\ A \xrightarrow{0.3} f[A],\ A \xrightarrow{0.7} (ab, cd)\}$, $J[(x_1, x_2)] = x_1x_2$ and $f[(x_1, x_2)] = (ax_1b, cx_2d)$. Then, $A \stackrel{*}{\Rightarrow} (ab, cd)$ with probability 0.7 by the third rule, which is followed by $A \stackrel{*}{\Rightarrow} f[(ab, cd)] = (aabb, ccdd)$ with probability $0.3 \cdot 0.7 = 0.21$

Table 4.1. SMCFG G_R

Type	Rule set	Function	Transition prob.	Emission prob.
E	$W_v \to (\varepsilon, \varepsilon)$		1	1
S	$W_v \to J[W_y]$	$J[(x_1, x_2)] = x_1x_2$	$t_v(y)$	1
D	$W_v \to SK[W_y]$	$SK[(x_1, x_2)] = (x_1, x_2)$	$t_v(y)$	1
$\mathrm{B_1}$	$W_v \to C_1[W_y, W_z]$	$C_1[x_1, (x_{21}, x_{22})] = (x_1x_{21}, x_{22})$	1	1
$\mathrm{B_2}$	$W_v \to C_2[W_y, W_z]$	$C_2[x_1, (x_{21}, x_{22})] = (x_{21}x_1, x_{22})$	1	1
$\mathrm{B_3}$	$W_v \to C_3[W_y, W_z]$	$C_3[x_1, (x_{21}, x_{22})] = (x_{21}, x_1x_{22})$	1	1
$\mathrm{B_4}$	$W_v \to C_4[W_y, W_z]$	$C_4[x_1, (x_{21}, x_{22})] = (x_{21}, x_{22}x_1)$	1	1
$\mathrm{U_{1L}}$	$W_v \to UP_{1L}^{a_i}[W_y]$	$UP_{1L}^{a_i}[(x_1, x_2)] = (a_ix_1, x_2)$	$t_v(y)$	$e_v(a_i)$
$\mathrm{U_{1R}}$	$W_v \to UP_{1R}^{a_j}[W_y]$	$UP_{1R}^{a_j}[(x_1, x_2)] = (x_1a_j, x_2)$	$t_v(y)$	$e_v(a_j)$
$\mathrm{U_{2L}}$	$W_v \to UP_{2L}^{a_k}[W_y]$	$UP_{2L}^{a_k}[(x_1, x_2)] = (x_1, a_kx_2)$	$t_v(y)$	$e_v(a_k)$
$\mathrm{U_{2R}}$	$W_v \to UP_{2R}^{a_l}[W_y]$	$UP_{2R}^{a_l}[(x_1, x_2)] = (x_1, x_2a_l)$	$t_v(y)$	$e_v(a_l)$
P	$W_v \to BP^{a_ia_l}[W_y]$	$BP^{a_ia_l}[(x_1, x_2)] = (a_ix_1, x_2a_l)$	$t_v(y)$	$e_v(a_i, a_l)$

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, $L(G_1) = \{a^nb^nc^nd^n \mid n \geq 1\}$. □

In this chapter, we focus on a $(2, 2)$-SMCFG $G_R = (N, T, F, P, S)$ with $\deg(G_R) \leq 5$ 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, $\mathrm{B_1}$, $\mathrm{B_2}$, $\mathrm{B_3}$, $\mathrm{B_4}$, $\mathrm{U_{1L}}$, $\mathrm{U_{1R}}$, $\mathrm{U_{2L}}$, $\mathrm{U_{2R}}$ or P (see Table 4.1). These types stand for END, START, DELETE, BIFURCATION, UNPAIR and PAIR respectively. The type of W_v is denoted by type(v) and we predefine type$(1) = \mathrm{S}$, that is, W_1 is the start symbol. For each rule r, two real values called *transition probability* p_1 and *emission probability* p_2 are specified as shown in Table 4.1. The probability of r is simply defined as $p_1 \cdot p_2$. In application, $p_1 = t_v(y)$ and $p_2 = e_v(a_i)$, etc. in Table 4.1 are parameters for the grammar, which are set by hand or by a training algorithm depending on the set of possible sequences to be analyzed. All the transition probabilities of bifurcation nonterminals are defined as one since most of the nonterminals for modeling RNA secondary structure have the type of either UNPAIR or PAIR, and BIFURCATION nonterminals are sometimes used to deal with concatenating and wrapping operation. This single choice of transition for BIFURCATION nonterminal reduces time complexities of SMCFG algorithms. For each nonterminal W_v, Δ_v^{1L}, Δ_v^{1R}, Δ_v^{2L} and Δ_v^{2R} are defined as the number of symbols generated by W_v (see Table 4.2). This notation simplifies

Table 4.2. The number of symbols emitted by nonterminals

Type	Δ_v^{1L}	Δ_v^{1R}	Δ_v^{2L}	Δ_v^{2R}
E	0	0	0	0
S	0	0	0	0
D	0	0	0	0
B_1	0	0	0	0
B_2	0	0	0	0
B_3	0	0	0	0
B_4	0	0	0	0
U_{1L}	1	0	0	0
U_{1R}	0	1	0	0
U_{2L}	0	0	1	0
U_{2R}	0	0	0	1
P	1	0	0	1

the description of the algorithms presented in the next section.

3. Algorithms for SMCFG

In RNA structure analysis using stochastic grammars, we have to deal with the following three problems [8]:

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

3.1 Alignment Algorithm

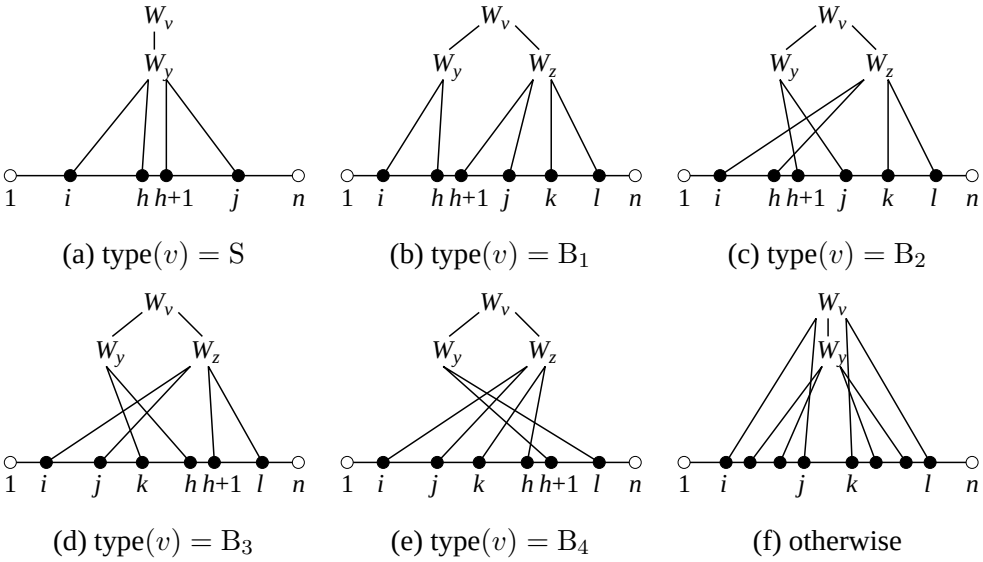
The alignment problem for G_R is to find the most probable derivation tree for a given input sequence. This problem can be solved by a dynamic programming algorithm similar to the CYK algorithm for SCFGs [8], and in this paper, we also call the parsing algorithm for G_R the CYK algorithm. We fix an input 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 maximum log 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 maximum log probabilities for an empty sequence ε and a pair of ε. 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 let $\mathcal{C}_v = \{y \mid W_v \to f[W_y] \in P,\ f \in F\}$. 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 γ, which leads to $\log P(w, \hat{\pi} \mid \theta)$ where $\hat{\pi}$ is the most probable derivation tree and θ is an entire set of probability parameters. The illustration of the iteration step in the CYK algorithm is shown in Figure 4.1. The detailed description of the algorithm is as follows:

Algorithm 4.1 (CYK)**.**
Initialization:

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

Iteration:

5 **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
6 **do if** type$(v) = \mathrm{E}$
7 **then if** $j = i-1$ **and** $l = k-1$
8 **then** skip
9 **else** $\gamma_v(i,j,k,l) \leftarrow -\infty$



(a) type$(v) = \mathrm{S}$ (b) type$(v) = \mathrm{B}_1$ (c) type$(v) = \mathrm{B}_2$

(d) type$(v) = \mathrm{B}_3$ (e) type$(v) = \mathrm{B}_4$ (f) otherwise

Figure 4.1. Illustration of the iteration step for calculating γ

if type$(v) = \mathrm{S}$

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$(v) = \mathrm{B}_1$ **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$(v) = \mathrm{B}_2$ **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$(v) = \mathrm{B}_3$ **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$(v) = \mathrm{B}_4$ **and** $W_v \to C_4[W_y, W_z]$

then $\gamma_v(i,j,k,l) \leftarrow \max_{h=k-1,\ldots,l} [\gamma_z(i,j,k,h) + \gamma_y(h+1,l)]$

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

if type$(v) = \mathrm{P}$

then if $j = i-1$ **or** $l = k-1$

27 **then** $\gamma_v(i,j,k,l) \leftarrow -\infty$

28 **else** $\gamma_v(i,j,k,l) \leftarrow \max_{y \in \mathcal{C}_v}[\log e_v(a_i,a_l) + \log t_v(y) + \gamma_y(i+1,j,k,l-1)]$

29 $\tau_v(i,j,k,l) \leftarrow \arg\max_{y}[\log e_v(a_i,a_l) + \log t_v(y) + \gamma_y(i+1,j,k,l-1)]$

30 **else** $\gamma_v(i,j,k,l) \leftarrow \max_{y \in \mathcal{C}_v}[\log e_v(a_i,a_j,a_k,a_l) + \log t_v(y)$
$+\gamma_y(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})]$

31 $\tau_v(i,j,k,l) \leftarrow \arg\max_{y}[\log e_v(a_i,a_j,a_k,a_l) + \log t_v(y)$
$+\gamma_y(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R})]$ □

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 τ 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 as follows:

Algorithm 4.2 (CYK traceback)**.**

Initialization:

$(v,h) \leftarrow \tau_1(1,n)$

attach W_v as the child of W_1

push $(v,1,h,h+1,n)$ on the stack

Iteration:

while the stack is not empty

do pop

if type$(v) = \mathrm{E}$

then attach $(\varepsilon,\varepsilon)$ as the child of W_v

if type$(v) = \mathrm{S}$

then $(y,h) \leftarrow \tau_v(i,j)$

attach W_y as the child of W_v

push $(y,i,h,h+1,j)$

if type$(v) = \mathrm{B}_1$

then $(y,z,h) \leftarrow \tau_v(i,j,k,l)$

attach W_y, W_z as the children of W_v

push $(z,h+1,j,k,l)$

push (y,i,h)

17 **if** type$(v) = \mathrm{B}_2$
18 **then** $(y, z, h) \leftarrow \tau_v(i, j, k, l)$
19 attach W_y, W_z as the children of W_v
20 push (z, i, h, k, l)
21 push $(y, h + 1, j)$
22 **if** type$(v) = \mathrm{B}_3$
23 **then** $(y, z, h) \leftarrow \tau_v(i, j, k, l)$
24 attach W_y, W_z as the children of W_v
25 push $(z, i, j, h + 1, l)$
26 push (y, k, h)
27 **if** type$(v) = \mathrm{B}_4$
28 **then** $(y, z, h) \leftarrow \tau_v(i, j, k, l)$
29 attach W_y, W_z as the children of W_v
30 push (z, i, j, k, h)
31 push $(y, h + 1, l)$
32 **else** $y \leftarrow \tau_v(i, j, k, l)$
33 attach W_y as the child of W_v
34 push $(y, i + \Delta_v^{1L}, j - \Delta_v^{1R}, k + \Delta_v^{2L}, l - \Delta_v^{2R})$ □

3.2 Scoring Algorithm

As in SCFGs [8], 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. When the calculation terminates, we obtain the likelihood $P(w \mid \theta) = \alpha_1(1, n)$ of the sequence w given the probability parameters θ. The time and space complexities of the algorithm are identical with those of the CYK algorithm.

Algorithm 4.3 (Inside)**.**
Initialization:

1 **for** $i \leftarrow 1$ **to** $n+1$, $k \leftarrow i$ **to** $n+1$, $v \leftarrow 1$ **to** m
2 **do if** type$(v) = \mathrm{E}$
3 **then** $\alpha_v(i, i-1, k, k-1) \leftarrow 1$
4 **else** $\alpha_v(i, i-1, k, k-1) \leftarrow 0$

Iteration:

5 **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
6 **do if** type$(v) = \mathrm{E}$
7 **then if** $j = i-1$ **and** $l = k-1$
8 **then** skip
9 **else** $\alpha_v(i, j, k, l) \leftarrow 0$
10 **if** type$(v) = \mathrm{S}$
11 **then** $\alpha_v(i, j) \leftarrow \sum_{y \in \mathcal{C}_v} \sum_{h=i-1}^{j} t_v(y) \alpha_y(i, h, h+1, j)$
12 **if** type$(v) = \mathrm{B}_1$
13 **then** $\alpha_v(i, j, k, l) \leftarrow \sum_{h=i-1}^{j} \alpha_y(i, h) \alpha_z(h+1, j, k, l)$
14 **if** type$(v) = \mathrm{B}_2$
15 **then** $\alpha_v(i, j, k, l) \leftarrow \sum_{h=i-1}^{j} \alpha_y(h+1, j) \alpha_z(i, h, k, l)$
16 **if** type$(v) = \mathrm{B}_3$
17 **then** $\alpha_v(i, j, k, l) \leftarrow \sum_{h=k-1}^{l} \alpha_z(i, j, h+1, l) \alpha_y(k, h)$
18 **if** type$(v) = \mathrm{B}_4$
19 **then** $\alpha_v(i, j, k, l) \leftarrow \sum_{h=k-1}^{l} \alpha_z(i, j, k, h) \alpha_y(h+1, l)$
20 **if** type$(v) = \mathrm{P}$
21 **then if** $j = i-1$ **or** $l = k-1$
22 **then** $\alpha_v(i, j, k, l) \leftarrow 0$
23 **else** $\alpha_v(i, j, k, l) \leftarrow \sum_{y \in \mathcal{C}_v} e_v(a_i, a_l) t_v(y) \alpha_y(i+1, j, k, l-1)]$
24 **else** $\alpha_v(i, j, k, l) \leftarrow \sum_{y \in \mathcal{C}_v} e_v(a_i, a_j, a_k, a_l) t_v(y)$
$\alpha_y(i + \Delta_v^{1L}, j - \Delta_v^{1R}, k + \Delta_v^{2L}, l - \Delta_v^{2R})$ □

In order to re-estimate the probability parameters of G_R, we need the outside algo-

rithm. 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 subsequences $(a_i \cdots a_j, a_k \cdots a_l)$. 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 β requires the inside variables α. 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. Figure 4.2 shows the iteration step in the algorithm. Formal description of the outside algorithm is as follows:

Algorithm 4.4 (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 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 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 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 type$(v) = \mathrm{S}$ **and** $W_y \to C_4[W_v, W_z]$

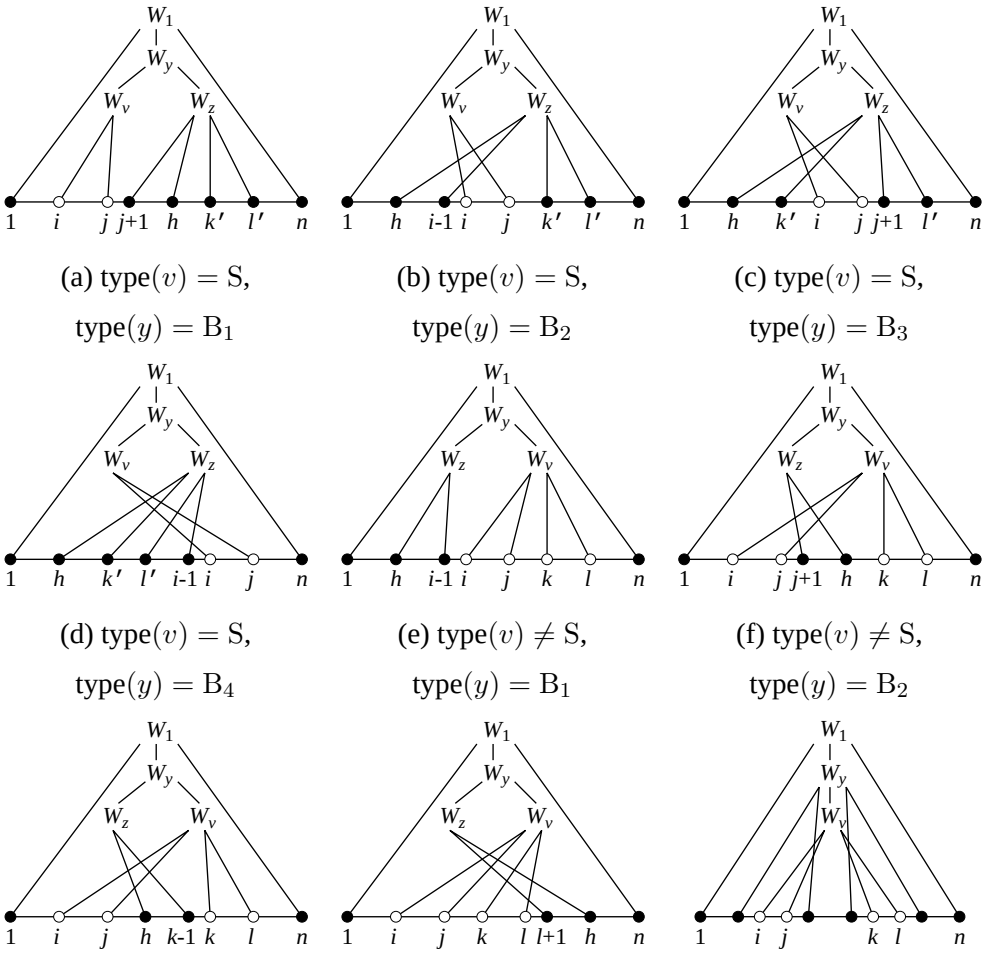
    then $\beta_v(i,j) \leftarrow \sum_{h=1}^{i} \sum_{k'=h-1}^{i-1} \sum_{l'=k'+1}^{i} \beta_y(h,k',l',j)\alpha_z(h,k',l',i-1)$

  if 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 type$(v) \neq \mathrm{S}$ **and** $W_y \to C_2[W_z, W_v]$

    then $\beta_v(i,j,k,l) \leftarrow \sum_{h=j}^{k-1} \beta_y(i,h,k,l)\alpha_z(j+1,h)$



(a) type$(v) = \mathrm{S}$, type$(y) = \mathrm{B}_1$ (b) type$(v) = \mathrm{S}$, type$(y) = \mathrm{B}_2$ (c) type$(v) = \mathrm{S}$, type$(y) = \mathrm{B}_3$

(d) type$(v) = \mathrm{S}$, type$(y) = \mathrm{B}_4$ (e) type$(v) \neq \mathrm{S}$, type$(y) = \mathrm{B}_1$ (f) type$(v) \neq \mathrm{S}$, type$(y) = \mathrm{B}_2$

(g) type$(v) \neq \mathrm{S}$, type$(y) = \mathrm{B}_3$ (h) type$(v) \neq \mathrm{S}$, type$(y) = \mathrm{B}_4$ (i) otherwise

Figure 4.2. Illustration of the iteration step for calculating β

15 **if** type$(v) \neq \mathrm{S}$ **and** $W_y \to C_3[W_z, W_v]$

16 **then** $\beta_v(i,j,k,l) \leftarrow \sum_{h=j+1}^{k} \beta_y(i,j,h,l)\alpha_z(h,k-1)$

17 **if** type$(v) \neq \mathrm{S}$ **and** $W_y \to C_4[W_z, W_v]$

18 **then** $\beta_v(i,j,k,l) \leftarrow \sum_{h=l}^{n} \beta_y(i,j,k,h)\alpha_z(l+1,h)$

19 **else** $\beta_v(i,j,k,l) \leftarrow \sum_{y \in \mathcal{P}_v} \beta_y(i-\Delta_y^{1L}, j+\Delta_y^{1R}, k-\Delta_y^{2L}, l+\Delta_y^{2R})$
$e_y(a_{i-\Delta_y^{1L}}, a_{j+\Delta_y^{1R}}, a_{k-\Delta_y^{2L}}, a_{l+\Delta_y^{2R}})t_y(v)$ □

3.3 Training Algorithm

The training problem for G_R can be solved by the EM algorithm called the inside-outside algorithm where the inside variables α and outside variables β 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 $w^{(s)}$ $(1 \leq s \leq N)$. Let $\alpha^{(s)}$ and $\beta^{(s)}$ be the inside and outside variables calculated for each input sequence $w^{(s)}$. Then we can obtain the expected number of times $E(v)$ that a nonterminal W_v is used for training sequences $w^{(s)}$ $(1 \leq r \leq N)$ by summing the above terms over all sequences (E-step):

$$E(v) = \begin{cases} \displaystyle\sum_{s=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \frac{1}{P(w^{(s)} \mid \theta)} \alpha_v^{(s)}(i,j)\beta_v^{(s)}(i,j), & (\text{type}(v) = \text{S}) \\ \displaystyle\sum_{s=1}^{N} \sum_{i=1}^{n+1} \sum_{j=i-1}^{n} \sum_{k=j+1}^{n+1} \sum_{l=k-1}^{n} \frac{1}{P(w^{(s)} \mid \theta)} \alpha_v^{(s)}(i,j,k,l) & \\ \quad \beta_v^{(s)}(i,j,k,l). & (\text{type}(v) \neq \text{S}) \end{cases}$$

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 as follows:

$$E(v \to y) = \begin{cases} \displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n+1}\sum_{j=i-1}^{n}\sum_{h=i-1}^{j} \frac{1}{P(w^{(s)} \mid \theta)}\beta_v^{(s)}(i,j)t_v(y) & \\ \quad \alpha_y^{(s)}(i,h,h+1,j), & (\text{type}(v) = \text{S}) \\ \displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n+1}\sum_{j=i-1}^{n}\sum_{k=j+1}^{n+1}\sum_{l=k-1}^{n} \frac{1}{P(w^{(s)} \mid \theta)}\beta_v^{(s)}(i,j,k,l) & \\ \quad e_v(a_i,a_j,a_k,a_l)t_v(y) & \\ \quad \alpha_y^{(s)}(i+\Delta_v^{1L}, j-\Delta_v^{1R}, k+\Delta_v^{2L}, l-\Delta_v^{2R}). & (\text{type}(v) \neq \text{S}) \end{cases}$$

For a given terminal a or a pair of terminals (a,b), 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 is

$$E(v \to a) = \begin{cases} \displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j+1}^{n+1}\sum_{l=k-1}^{n} \frac{1}{P(w^{(s)} \mid \theta)}\delta(a_i^{(s)} = a) & \\ \quad \beta_v^{(s)}(i,j,k,l)\alpha_v^{(s)}(i,j,k,l), & (\text{type}(v) = \text{U}_{1\text{L}}) \\ \displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j+1}^{n+1}\sum_{l=k-1}^{n} \frac{1}{P(w^{(s)} \mid \theta)}\delta(a_j^{(s)} = a) & \\ \quad \beta_v^{(s)}(i,j,k,l)\alpha_v^{(s)}(i,j,k,l), & (\text{type}(v) = \text{U}_{1\text{R}}) \\ \displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n-1}\sum_{j=i-1}^{n-1}\sum_{k=j+1}^{n}\sum_{l=k}^{n} \frac{1}{P(w^{(s)} \mid \theta)}\delta(a_k^{(s)} = a) & \\ \quad \beta_v^{(s)}(i,j,k,l)\alpha_v^{(s)}(i,j,k,l), & (\text{type}(v) = \text{U}_{2\text{L}}) \\ \displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n-1}\sum_{j=i-1}^{n-1}\sum_{k=j+1}^{n}\sum_{l=k}^{n} \frac{1}{P(w^{(s)} \mid \theta)}\delta(a_l^{(s)} = a) & \\ \quad \beta_v^{(s)}(i,j,k,l)\alpha_v^{(s)}(i,j,k,l), & (\text{type}(v) = \text{U}_{2\text{R}}) \end{cases}$$

and for $\text{type}(v) = \text{P}$,

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

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

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 (M-step):

$$\hat{t}_v(y) = \frac{E(v \to y)}{E(v)},\ \hat{e}_v(a) = \frac{E(v \to a)}{E(v)},\ \hat{e}_v(a,b) = \frac{E(v \to ab)}{E(v)}. \tag{4.1}$$

Note that the expected count correctly corresponding to its nonterminal type must be substituted for the above equations. For example, if type$(v) = \mathrm{S}$,

$$\hat{t}_v(y) = \frac{\displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n+1}\sum_{j=i-1}^{n}\sum_{h=i-1}^{j} \frac{1}{P(w^{(s)} \mid \theta)}\beta_v^{(s)}(i,j)t_v(y)\alpha_y^{(s)}(i,h,h+1,j)}{\displaystyle\sum_{s=1}^{N}\sum_{i=1}^{n+1}\sum_{j=i-1}^{n} \frac{1}{P(w^{(s)} \mid \theta)}\alpha_v^{(s)}(i,j)\beta_v^{(s)}(i,j)}.$$

In summary, the inside-outside algorithm is as follows:

Algorithm 4.5 (Inside-Outside)**.**

Initialization:

1 Pick arbitrary probability parameters of the model.

Iteration:

2 Calculate the new probability parameters using (4.1).

3 Calculate the new log likelihood $\sum_{s=1}^{N} \log P(w^{(s)} \mid \theta)$ of the model.

Termination:

4 Stop if the change in log likelihood is less than predefined threshold. □

3.4 Scanning Algorithm

Finally, we mention another application using the extension of the CYK algorithm. The extended CYK algorithm can be used to find ncRNA genes in long genome sequences. The basic idea is that the most likely parsing is done in a target window (subsequence) of variable length and then the window is moved along the whole sequence. However, the standard CYK algorithm described before does not work well since the maximum log probability of the derivation of a subsequence strongly depends on its length. To avoid such a phenomenon, we calculate log odds (LOD) scores that are the ratio to a

null model generating random sequences. Assume that the likelihood of a sequence generated by the null model is the product of each base frequence denoted by f_a and so on. In the scanning algorithm below, we will use the following score $\hat{e}_v$ instead of log emission probability e_v for nonterminal types of $\mathrm{U_{1L}}$, $\mathrm{U_{1R}}$, $\mathrm{U_{2L}}$, $\mathrm{U_{2R}}$ and P:

$$\hat{e}_v(a_i, a_j, a_k, a_l) = \log \frac{e_v(a_i, a_j, a_k, a_l)}{f_{a_i} f_{a_j} f_{a_k} f_{a_l}}$$

Note that if type$(v) = \mathrm{U_{1L}}$, $\hat{e}_v(a_i, a_j, a_k, a_l) = \hat{e}_v(a_i) = \log(e_v(a_i)/f_{a_i})$. The other types are interpreted similarly. Let r denote the right end of the target region, and $d_{\min}$ and $d_{\max}$ denote the minimum and the maximum lengths of the target window respectively. Also, n denotes the length of an input genome sequence, and the indices i, j, k and l are interpreted as in the CYK algorithm.

Algorithm 4.6 (CYK-scan)**.**

1 **for** $r \leftarrow d_{\min}$ **to** n, $d \leftarrow d_{\min}$ **to** $d_{\max}$, $i \leftarrow r$ **downto** $r - d + 1$, $j \leftarrow i - 1$ **to** r, $k \leftarrow r + 1$ **downto** $j + 1$, $l \leftarrow k - 1$ **to** r

2 **do** CYK algorithm □

The time complexity is $O(amnd_{\max}^4 + bnd_{\max}^5)$ where b is the number of BIFURCATION nonterminals, a is the number of other nonterminals and $m = a + b$. The space complexity is $O(md_{\max}^4)$.

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) [10] that contains multiple sequence alignments and covariance models [9] 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) for prediction tests (see Table 4.3). Corona_pk3 has a simple pseudoknotted structure, whereas HDV_ribozyme and Tombus_3_IV have more complicated structures with pseudoknots. Also, we used several genome sequences known to have Corona_pk3 genes to test ncRNA gene finding, which are also available from Rfam.

Table 4.3. Three RNA families from Rfam ver. 7.0

Family	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

4.2 Implementation

We specified a particular SMCFG G_R by utilizing secondary structure annotation of each family in Rfam. Rules were determined by considering consensus secondary structure. Probability parameters were estimated in a few selected sequences by the simplest pseudocounting method known as the Laplace's rule [8]: to add one extra count to the true counts for each base configuration observed in the 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 4.3).

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 $O(mn^4)$). Since the dynamic programming matrix in our specified model is sparse, we successfully implemented the matrix as a *hash table* storing only nonzero probability values (equivalently, finite values of the logarithm of probabilities). Consider the case where both of the number of nonterminals and the sequence length is 100. If we try to implement the DP matrix representing $\gamma_v(i, j, k, l)$ as a five dimensional array, about 200 GB memory space will be required. On the other hand, using a hash table of the size 2^{17}, only 5.8 MB memory space will be sufficient for implementing γ. This point deserves explicit emphasis for computational experiments.

4.3 Structure Prediction

We tested prediction accuracy by calculating precision and recall, 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 respec-

Table 4.4. Prediction results

Family	Precision [%]			Recall [%]			CPU time [sec]		
	Avg	Min	Max	Avg	Min	Max	Avg	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
    [[[[[[[[                ((((((((((   ]]]]]]]]    ((((((((((
```

Figure 4.3. Comparison between a trusted structure and a predicted one

tively. The results are shown in Table 4.4. A nearly correct prediction (94.4% precision and recall) for Corona_pk3 is shown in Figure 4.3 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 $O(m^2n^4)$.

To see whether a grammar for one RNA family can be applied to secondary structure prediction for another family, we compared LOD scores obtained by applying the specific grammar for Corona_pk3 to all of the three RNA families (see Table 4.5). As a result, we can say that a specific grammar overfits its objective RNA family.

4.4 Comparison with PSTAG

We compared the prediction accuracy of our SMCFG algorithm with that of the PSTAG algorithm [20] (see Table 4.6). 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. The PSTAG algorithm, based

Table 4.5. Comparison of LOD scores by using the same grammar

Family	Grammar	# of test sequences	LOD score [bit]		
			Avg	Min	Max
Corona_pk3	Corona_pk3	10	38.2	7.2	72.5
HDV_ribozyme	Corona_pk3	10	−6.4	−12.3	−2.7
Tombus_3_IV	Corona_pk3	12	−3.5	−5.5	−0.3

Table 4.6. 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

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. As the table shows, the SMCFG method is at least comparable to the PSTAG method in the same test sets.

4.5 Detection of Non-Coding RNA Gene

We did elementary gene finding tests using the CYK-scan algorithm. Figure 4.4 shows the result of Corona_pk3 gene finding in genome sequence X90577 of length 1137. In the figure, scores are plotted at the right end of the target window. The region with the highest score is 1000–1062, which is exactly the same as the correct region. To see whether the scanning algorithm can distinguish a real ncRNA gene from a shuffled gene, we also tested a sequence where the real gene is replaced with a shuffled one. The shuffling procedure we used is the *dinucleotide shuffling* [3] that preserves dinucleotide frequency of an input sequence. As might be expected, the hits of the high-scoring subsequences disappeared after shuffling (see Figure 4.5). Tests for other genome sequences such as X90576 and X51325 of length 946 and 1576, respectively, similarly showed strong scores in each correct region of the original sequences and low scores in each shuffled region.

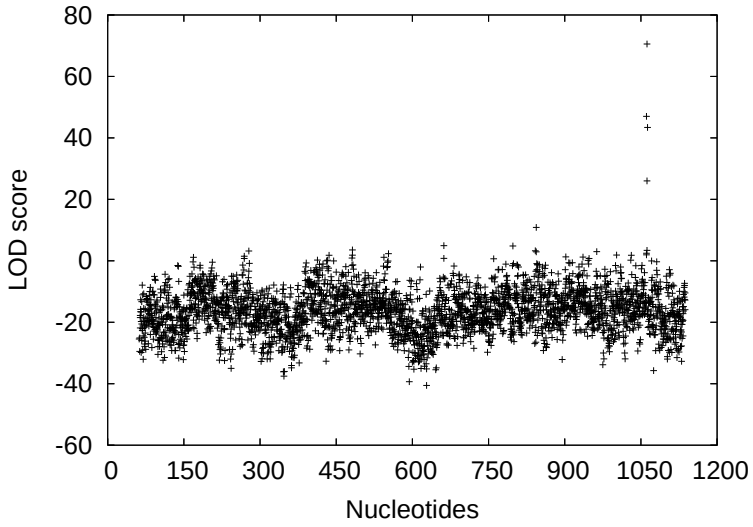


Figure 4.4. Gene detection in genome sequence X90577

5. Discussion

In the computational experiments using the SMCFG algorithm, we obtained good prediction results in terms of accuracy, and we did not trained probability parameters using the inside-outside algorithm any more. Main 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 the consensus structure and the number of gaps in each alignment is relatively few. In fact, an earlier experimental results, omitted in this thesis, 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 4.4. 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 the alignment.

The PSTAG method aligns an unfolded sequence with a derivation tree representing trusted structure. In SMCFGs, rules are constructed according to a consensus structure and then the most likely derivation tree is calculated. In this sense, SMCFGs and PSTAGs have a common property that both of them take structural alignment into

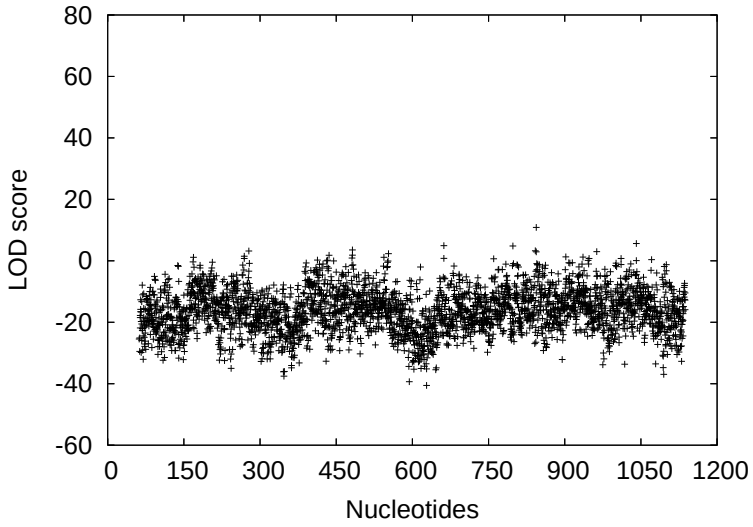


Figure 4.5. Gene detection in X90577 with shuffled gene

consideration implicitly or explicitly. The time and space complexities of the SMCFG algorithm have the same order as those of the PSTAG algorithm, whereas the SMCFG algorithm consumes less memory than the PSTAG algorithm since the dynamic programming matrix of the 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 SMCFGs and PSTAGs 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, the PSTAG method takes three finite states into account, which represent match, insertion and deletion states. Here, we regard nonterminals as states and rule application as state transitions [8]. The difference of the number of finite states may affect prediction accuracy.

Rivas and Eddy [25] compared the scores of an original genome sequence known to have ncRNA genes with those of a sequence including mononucleotide shuffled genes, which contain the same mononucleotide frequency as that of the genes. In their paper, they concluded that it would not be statistically significant to use secondary structure as a signal for detecting ncRNA genes. However, as our experimental results indicate, the dinucleotide shuffling is the key to test the significance, and we can say that it will be useful to use secondary structure as a statistical signal for a certain type of gene

finding tests.

6. Summary

In this chapter, we proposed a probabilistic model named SMCFG, and designed a polynomial time parsing and a parameter estimation algorithm for the specific SMCFG. We then carried out computational experiments on RNA secondary structure prediction with pseudoknots using the SMCFG parsing algorithm. The results of the experiments indicated good performance of the algorithm in terms of accuracy. Also, we can say that the SMCFG method is at least comparable to the PSTAG method in the same test sets. Moreover, elementary tests on ncRNA gene finding showed good results for a family of ncRNAs with pseudoknots.

Comparing with other prediction methods such as a thermodynamic approach, stochastic grammars have an advantage in easily modeling RNA secondary structure that 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.

Chapter 5

Conclusion

This thesis dealt with several formal grammars for describing RNA secondary structure including pseudoknots and their application to structure analysis. In particular, the following subjects were discussed:

(1) Comparison of the generative power of grammars for describing RNA pseudoknotted structure.

(2) Analysis of RNA pseudoknotted structure by using a stochastic grammar-based approach.

Multiple context-free grammars played an important role in both (1) and (2).

In Chapter 3, we dealt with the first subject. Namely, the classes of grammars SLTAGs, ESLTAGs and RPGs for representing RNA pseudoknotted structure were identified as subclasses of MCFGs, and their generative powers were compared. Considering the results obtained in Chapter 3, we conclude that ESLTAGs (equivalently, ESLMCFGs or $(2, 2)$-MCFGs with degree 5 or less) are candidates for the minimum grammars that can represent pseudoknots. We also showed that $\mathcal{SLTAL}$ is a full trio and $\mathcal{ESLTAL}$ is a substitution closed full AFL, which is a good property from the formal language theoretical point of view.

To address the second subject, we proposed a probabilistic model named SMCFG in Chapter 4, where the subclass of MCFGs corresponding to the class of ESLTAGs was extended to a stochastic grammar. We designed a polynomial time parsing and a parameter estimation algorithm for the specific SMCFG. We then carried out computational experiments on RNA secondary structure prediction with pseudoknots using the

SMCFG parsing algorithm. The results of the experiments indicated good performance of the algorithm in terms of accuracy. Also, we can say that the SMCFG method is at least comparable to the PSTAG method in the same test RNA sequences. Moreover, elementary tests on ncRNA gene finding showed good results for a family of ncRNAs with pseudoknots.

The generative power of a grammar and computational complexity for parsing have a close relationship. In this sense, it is reasonable to identify an optimal class of grammars for RNA and predict secondary structure by using the grammar. The proposed prediction algorithm for SMCFGs needs $O(n^5)$ computation time as well as existing algorithms for ESLTAGs and PSTAGs, where n is the length of an input sequence. Although this complexity seems very high at a glance, the experiments we performed make it clear that prediction can be performed in a practical time for RNA sequences of relatively short length.

We predicted known RNA secondary structure by using the parsing algorithm of a specific grammar for its corresponding RNA family. A specific model can be useful for discrimination between the objective family and other families. The reason is that a prediction algorithm shows a strong score for a sequence that can be generated by the grammar designed for the corresponding RNA structure as compared with a sequence that is unlikely to be generated by the grammar. For prediction of unknown structure, we need a generic grammar for which the accuracy of its prediction algorithm has to be good to some extent. The construction of a generic model is difficult since there are various kinds of RNA structures. However, if a family of RNA families (i.e., superfamily) whose structures are partially similar to each other is specified, we may construct a generic grammar. For example, since transition probabilities for nonterminals generating consecutive base pairs determine the probability distribution of the length of a stem, setting the transition probabilities carefully may contribute to the construction of a generic grammar. Although the probability parameters of the rules can be trained in our approach, the rules of the grammar themselves cannot be changed any more. Considering from this point of view, it is important to automatically derive an optimal grammar from sequence data with structure, rather than to fix an initial grammar.

In this thesis, we compared the yield languages generated by the grammars for RNA, but we should notice that a secondary structure is represented by a derivation (or derived) tree. Thus, it would be more important to compare each tree language

rather than each yield language. That is, a comparison of the tree generative power of ESLTAGs and RPGs is an interesting problem. Turning now to structure prediction of biomolecules, Abe and Mamitsuka [1] used a subclass of stochastic tree grammars called stochastic ranked node rewriting grammars (SRNRGs) for predicting protein secondary structure. SRNRGs have enough generative power to deal with anti-parallel and parallel dependency, and combinations of them in β-sheets, which is more difficult to handle than RNA secondary structure. The major difference between MCFGs and other grammars fully describing RNA pseudoknots is that the generative power of MCFGs can be set arbitrarily by choosing grammar parameters: dimension, rank and degree. Thus, the SMCFG method can be applied to more complicated biological structure analysis such as predicting protein secondary structure.

References

[1] N. Abe and H. Mamitsuka, “Predicting protein secondary structure using stochastic tree grammars,” Machine Learning, vol. 29, pp. 275–301, 1997.

[2] T. Akutsu, “Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots,” Discrete Applied Mathematics, vol. 104, pp. 45–62, 2000.

[3] S. F. Altschul and B. W. Erickson, “Significance of nucleotide sequence alignments: a method for random sequence permutation that preserves dinucleotide and codon usage,” Molecular Biology and Evolution, vol. 2, pp. 526–538, 1985.

[4] F. H. D. van Batenburg, A. P. Gultyaev, C. W. A. Pleij, J. Ng and J. Oliehoek, “PseudoBase: a database with RNA pseudoknots,” Nucleic Acids Research, vol. 28, no. 1, pp. 201–204, 2000.

[5] M. Brown and C. Wilson, “RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search,” Proceedings of Pacific Symposium on Biocomputing, pp. 109–125, 1996.

[6] L. Cai, R. L. Malmberg and Y. Wu, “Stochastic modeling of RNA pseudoknotted structures: a grammatical approach,” Bioinformatics, vol. 19, supp. 1, pp. i66–i73, 2003.

[7] A. Condon, “Problems on RNA secondary structure prediction and design,” 30th International Colloquium on Automata, Languages and Programming (ICALP2003), Lecture Notes in Computer Science, vol. 2719, pp. 22–32, 2003.

[8] R. Durbin, S. R. Eddy, A. Krogh and G. Mitchison, Biological Sequence Analysis, Cambridge University Press, 1998.

[9] S. R. Eddy and R. Durbin, “RNA sequence analysis using covariance models,” Nucleic Acids Research, vol. 22, no. 11, pp. 2079–2088, 1994.

[10] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna and S. R. Eddy, “Rfam: an RNA family database,” Nucleic Acids Research, vol. 31, no. 1, pp. 439–441, 2003.

[11] A. K. Joshi, L. Levy and M. Takahashi, “Tree adjunct grammars,” Journal of Computer & System Sciences, vol. 10, no. 1, pp. 136–163, 1975.

[12] A. K. Joshi and Y. Schabes, “Tree adjoining grammars,” in Handbook of Formal Languages, eds. G. Rozenberg and A. Salomaa, vol. 3 (Beyond Words), pp. 69–123, Springer, 1997.

[13] A. K. Joshi, K. Vijay-Shanker and D. J. Weir, “The convergence of mildly context-sensitive grammar formalisms,” Institute for Research in Cognitive Science, University of Pennsylvania, 1988.

[14] T. Kasami, H. Seki and M. Fujii, “Generalized context-free grammar and multiple context-free grammar,” IEICE Transactions on Information & Systems, vol. J71-D, no. 5, pp. 758–765, 1988 (in Japanese).

[15] T. Kasami, H. Seki and M. Fujii, “On the membership problem for head languages and multiple context-free languages,” IEICE Transactions on Information & Systems, vol. J71-D, no. 6, pp. 935–941, 1988 (in Japanese).

[16] Y. Kato, H. Seki and T. Kasami, “On the generative power of grammars for RNA secondary structure,” IEICE Transactions on Information & Systems, vol. E88-D, no. 1, pp. 53–64, 2005.

[17] Y. Kato, H. Seki and T. Kasami, “RNA pseudoknotted structure prediction using stochastic multiple context-free grammar,” IPSJ Transactions on Bioinformatics, vol. 47, no. SIG17 (TBIO1), pp. 12–21, 2006.

[18] K. Lari and S. J. Young, “The estimation of stochastic context-free grammars using the inside-outside algorithm,” Computer Speech and Language, vol. 4, pp. 35–56, 1990.

[19] A. Mateescu and A. Salomaa, “Aspects of classical language theory,” in Handbook of Formal Languages, eds. G. Rozenberg and A. Salomaa, vol. 1 (Word, Language, Grammar), pp. 175–251, Springer, 1997.

[20] H. Matsui, K. Sato and Y. Sakakibara, “Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures,” Bioinformatics, vol. 21, no. 11, pp. 2611–2617, 2005.

[21] O. Rambow and G. Satta, “A two-dimensional hierarchy for parallel rewriting systems,” IRCS Report 94-02, Institute for Research in Cognitive Science, University of Pennsylvania, 1994.

[22] O. Rambow and G. Satta, “Independent parallelism in finite copying parallel rewriting systems,” Theoretical Computer Science, vol. 223, pp. 87–120, 1999.

[23] E. Rivas and S. R. Eddy, “A dynamic programming algorithm for RNA structure prediction including pseudoknots,” Journal of Molecular Biology, vol. 285, pp. 2053–2068, 1999.

[24] E. Rivas and S. R. Eddy, “The language of RNA: a formal grammar that includes pseudoknots,” Bioinformatics, vol. 16, no. 4, pp. 334–340, 2000.

[25] E. Rivas and S. R. Eddy, “Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs,” Bioinformatics, vol. 16, no. 7, pp. 583–605, 2000.

[26] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood and D. Haussler, “Stochastic context-free grammars for tRNA modeling,” Nucleic Acids Research, vol. 22, pp. 5112–5120, 1994.

[27] Y. Sakakibara and C. Ferretti, “Splicing on tree-like structures,” Theoretical Computer Science, vol. 210, pp. 227–243, 1999.

[28] G. Satta and W. Schuler, “Restrictions on tree adjoining languages,” Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL98), pp. 1176–1182, 1998.

[29] H. Seki, T. Matsumura, M. Fujii and T. Kasami, “On multiple context-free grammars,” Theoretical Computer Science, vol. 88, pp. 191–229, 1991.

[30] 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.

[31] K. Vijay-Shanker, D. J. Weir and A. K. Joshi, “Tree adjoining and head wrapping,” Proceedings of the 11th International Conference on Computational Linguistics (COLING86), pp. 202–207, 1986.

[32] K. Vijay-Shanker, A study of tree adjoining grammar, Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, pp. 18–27, 1987.

Appendix

List of Abbreviations

Below is a table that indicates abbreviations used in this thesis.

Table 6.1. Abbreviations

Abbreviation	Formal name
AFL	abstract family of languages
CFG	context-free grammar
CIG	crossed-interaction grammar
CSG	context-sensitive grammar
ESLTAG	extended simple linear tree adjoining grammar
MCFG	multiple context-free grammar
PSTAG	pair stochastic tree adjoining grammar
RPG	RNA pseudoknot grammar
SCFG	stochastic context-free grammar
SMCFG	stochastic multiple context-free grammar
TAG	tree adjoining grammar
SLTAG	simple linear tree adjoining grammar
SSTAG	Satta and Schuler’s subclass of tree adjoining grammars