WFST に基づく確率文脈自由文法およびその拡張文法の高速 EM 学習法

亀谷　由隆[†]　森　高志[†]　佐藤　泰介[†]

現在，統計的言語モデルの一クラスとして確率文脈自由文法 (PCFG) が広く知られている．また，括弧なしコーパスから PCFG を訓練する方法として Inside-Outside (I-O) アルゴリズムが知られてきた．I-O アルゴリズムは PCFG 用に効率化を施した EM (expectation-maximization) アルゴリズムだが，依然その計算速度に問題があることが知られている．本論文では，文法構造があらかじめ与えられていることを前提に，訓練過程を構文解析と EM 学習に分離した高速 EM 学習法を提案する．その中間データ構造にパーザが生成する WFST (well-formed substring table) を用いる．例えば，一般化 LR パーザを用いると事前コンパイル・ボトムアップ探索による効率性，および Chomsky 標準形を要求しないという一般性を引き継ぐことができる．一方 EM 学習では，WFST のコンパクトさを利用して効率的なパラメタ推定が行なわれる．推定結果は I-O アルゴリズムで得られるものと一致する．更に，文脈依存性を取り入れた PCFG の拡張モデルに対する多項式オーダの EM 学習法を示す．また，ATR 対話コーパスを用いて実験を行ない，訓練時間が大幅に短縮されていることを確認した．

キーワード:　確率文脈自由文法，*EM* アルゴリズム，*Inside-Outside* アルゴリズム，*WFST*

Efficient EM learning of probabilistic CFGs and their extensions by using WFSTs

Yoshitaka KAMEYA[†] , Takashi MORI[†] and Taisuke SATO[†]

Probabilistic context-free grammars (PCFGs) are a widely-known class of statistical language models. The Inside-Outside (I-O) algorithm is also well-known as an efficient EM algorithm tailored for PCFGs. Although the algorithm requires only inexpensive linguistic resources, there remains a problem in its efficiency. In this paper, we present a new framework for efficient EM learning of PCFGs in which the parser is separated from the EM algorithm, assuming the underlying CFG is given. A new EM procedure exploits the compactness of WFSTs (well-formed substring tables) generated by the parser. Our framework is quite general in the sense that the input grammar need not to be in Chomsky normal form (CNF) while the new EM algorithm is equivalent to the I-O algorithm in the CNF case. In addition, we propose a polynomial-time EM procedure for CFGs with context-sensitive probabilities, and report experimental results with ATR corpus and a hand-crafted Japanese grammar.

KeyWords:　*Probabilistic context-free grammars, The EM algorithm, The Inside-Outside algorithm, Well-formed substring table*

† 東京工業大学 大学院情報理工学研究科 計算工学専攻, Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

1　はじめに

現在，統計的言語モデルの一クラスとして確率文脈自由文法（probabilistic context-free grammar; 以下 PCFG）が広く知られている．PCFG は文脈自由文法（context-free grammar; 以下 CFG）の生成規則に確率パラメタが付与されたものと見ることができ，それらのパラメタによって生成される文の確率が規定される．しかし，すべてのパラメタを人手で付けるのはコストと客観性の点で問題がある．そこで，計算機によるコーパスからの PCFG のパラメタ推定，すなわち PCFG の訓練 (training) が広く行なわれている．

現在，構造つきコーパス中の規則出現の相対頻度に基づき PCFG を訓練する方法（以下，相対頻度法と呼ぶ）が広く行なわれているが，我々はより安価な訓練データとして，分かち書きされている（形態素解析済みの）括弧なしコーパスを用いる．括弧なしコーパスからの PCFG の訓練法としては，Inside-Outside アルゴリズム (Baker 1979; Lari and Young 1990) が広く知られている（以下, I-O アルゴリズムと略す）．I-O アルゴリズムは CYK (Cocke-Younger-Kasami) パーザで用いられる三角行列の上に構築された，PCFG 用の EM (expectation-maximization) アルゴリズム (Dempster, Laird, and Rubin 1977) と特徴づけることができる．I-O アルゴリズムは多項式オーダの EM アルゴリズムであり，効率的とされているが，訓練コーパスの文の長さに対し 3 乗の計算時間を要するため，大規模な文法・コーパスからの訓練は困難であった．また，基になる CFG が Chomsky 標準形でなければならないという制約をもっている．

一方，本論文では，PCFG の文法構造（基になる CFG）が所与であるときの効率的な EM 学習法を提案する．提案手法は well-formed substring table（以下 WFST）と呼ばれるデータ構造を利用しており，全体の訓練過程を次の 2 段階に分離して PCFG を訓練する．

構文解析:

はじめにパーザによって与えられたテキストコーパスもしくはタグ付きコーパス中の各文に構文解析を施し，その文の構文木すべてを得る．ただし，構文木は実際に構築せずに途中で構築される WFST のままでとどめておく．

EM 学習:

上で得られた WFST から支持グラフと呼ばれるデータ構造を抽出し，新たに導出されたグラフィカル EM （graphical EM; 以下 gEM と略記）アルゴリズムを支持グラフ上で走らせる．

WFST は構文解析途中の部分的な解析結果（部分構文木）を格納するデータ構造の総称であり (田中 1988; 永田 1999)，パーザは WFST を参照することにより再計算を防いでいる．また，最終的に WFST に格納されている部分構文木を組み合わせて構文木を出力する．表 1 に各構文解析手法における WFST を掲げる．なお，Fujisaki らも文法が所与であるとして，上の 2 段階で PCFG を訓練する方法を提案しているが (Fujisaki, Jenelik, Cocke, Black, and Nishino 1989)，その方法では WFST は活用されていない．

提案手法の特長は従来法である I-O アルゴリズムの一般化と高速化が同時に実現された点，すなわち

特長 1: 従来の PCFG の EM 学習法の一般化となっている，

特長 2: 現実的な文法に対しては I-O アルゴリズムに比べて EM 学習が大幅に高速化される，

特長 3: 提案手法が，PCFG に文脈依存性を導入した確率言語モデル（PCFG の拡張文法[1] と呼ぶ）に対する多項式オーダの EM アルゴリズムを包含する

点にある．先述したように，I-O アルゴリズムは CYK 法の WFST である三角行列を利用して効率的に訓練を行なう手法と捉えることができ，提案手法の CYK 法と gEM アルゴリズムを組み合わせた場合が I-O アルゴリズムに対応する．一方，提案手法で Earley パーザや一般化 LR（以下 GLR）パーザと組み合わせる場合，文法構造に Chomsky 標準形を前提としないため，本手法は I-O アルゴリズムの一般化となっている（**特長 1**）．加えて，本論文では Stolcke の確率的 Earley パーザ (Stolcke 1995) や，Pereira と Schabes によって提案された括弧なしコーパスからの学習法 (Pereira and Schabes 1992) も提案手法の枠組で扱うことができる[2] ことを示す．また，**特長 2** が得られるのは，提案手法ではが WFST というコンパクトなデータ構造のみを走査するためである．そして，LR 表へのコンパイル・ボトムアップ解析といった特長により実用的には最も効率的とされる一般化 LR 法 (Tomita and Ng 1991)（以下 GLR 法）を利用できる点も訓練時間の軽減に効果があると考えられる．そして**特長 3** は提案手法の汎用性を示すものであり，本論文では北らの規則バイグラムモデル (Kita et al. 1994) の多項式オーダの EM アルゴリズムを提示する．

本論文の構成は次の通りである．まず節 2 で PCFG，CYK パーザ，I-O アルゴリズム，およびそれらの関連事項の導入を行なう．I-O アルゴリズムと対比させるため，提案手法を CYK パーザと gEM アルゴリズムの組合せを対象にした場合を節 3 で記述した．**特長 2** を検証するため，GLR パーザと gEM アルゴリズムを組み合わせた場合の訓練時間を ATR 対話コーパス (SLDB) を用いて計測した．その結果を節 4 に示す．また，**特長 3** を具体的に示すため，節 5 では PCFG の拡張文法に対する多項式オーダの EM アルゴリズムを提示する．最後に節 6 で関連研究について述べ，**特長 1** について考察する．本論文で用いる例文法，例文，およびそれ

1 Magerman らが (Magerman and Weir 1992) で述べている “Context-free grammar with context-sensitive probability (CFG with CSP)” を指す．具体的には Charniak らの疑似確率文脈依存文法 (Charniak and Carroll 1994) や北らの規則バイグラムモデル (Kita, Morimoto, Ohkura, Sagayama, and Yano 1994) が挙げられる．

2 より正確には，文法構造が与えられている場合の Pereira と Schabes の学習法を扱う．

表 1　各パーザにおける WFST．

パーザ	WFST
CYK 法	三角行列
Earley 法	アイテム集合（Earley チャート）の集まり
GLR 法	圧縮共有構文森 (packed shared parse forest)

らに基づく構文解析結果の多くは (永田 1999) のもの，もしくはそれに手を加えたものである．

以降では $A, B, \ldots$ を非終端記号を表すメタ記号，$a, b, \ldots$ を終端記号を表すメタ記号，ρ を一つの終端または非終端記号を表すメタ記号，ζ, ξ, ν を空列もしくは終端記号または非終端記号から成る記号列を表すメタ記号とする．空列は ε と書く．一方，一部の図を除き，具体的な文法記号を $\mathtt{S}, \mathtt{NP}, \ldots$ などタイプライタ書体で表す．また，y_n を第 n 要素とするリストを $\langle y_1, y_2, \ldots \rangle$ で表現する．またリスト $Y = \langle \ldots, y, \ldots \rangle$ であるとき，$y \in Y$ と書く．集合 X の要素数，記号列 ζ に含まれる記号数，リスト Y の要素数をそれぞれ $|X|$, $|\zeta|$, $|Y|$ で表す．これらはどれも見た目は同じだが文脈で違いを判断できる．

2　準備

2.1　確率文脈自由文法

はじめに，文脈自由文法 G を 4 つ組 $\langle V_{\mathrm{n}}, V_{\mathrm{t}}, R, S \rangle$ で定義する．ただし，V_{n} は非終端記号の集合，V_{t} は終端記号の集合，R は規則の集合，S は開始記号 ($S \in V_{\mathrm{n}}$) である．R 中の各規則 r は $A \to \zeta$ の形をしており，記号列 ζ' 中に非終端記号 A が出現するとき，A を ζ に置き換えることが可能であることを表す．我々は常に最左の非終端記号を置き換えるように規則を適用する（最左導出に固定する）．規則 r の適用により記号列 ζ が ξ に置き換えられるとき $\zeta \overset{r}{\Rightarrow} \xi$ と書く．このような置き換えを 0 回以上行なって ζ から ξ が得られるとき，$\zeta \overset{*}{\Rightarrow} \xi$ と書く．特に，置換えが 1 回以上であることを強調する場合は $\zeta \overset{+}{\Rightarrow} \xi$ と書く．S から導出可能な非終端記号列 $\mathbf{w}$ を文と呼ぶ．CFG G における文の集合を G の言語と呼び，L_G と書く．

そして，G に基づく PCFG を $G(\theta)$ で表す．逆に G を「PCFG $G(\theta)$ の文法構造」と呼ぶ．θ は $|R|$ 次元ベクトルであり，以降パラメタと呼ぶ．θ の各要素は $\theta(r)$ で参照され，$0 \leq \theta(r) \leq 1$, $\sum_{\zeta:(A \to \zeta) \in R} \theta(A \to \zeta) = 1$ が成り立つとする ($A \in V_{\mathrm{n}}$, $r \in R$)．PCFG では「適用する規則は他に影響を受けずに選択される」と仮定される．従って $\zeta_0 \overset{r_1}{\Rightarrow} \zeta_1 \overset{r_2}{\Rightarrow} \zeta_2 \overset{r_3}{\Rightarrow} \cdots \overset{r_K}{\Rightarrow} \zeta_K$ における規則の適用列 $\mathbf{r} = \langle r_1, r_2, \ldots, r_K \rangle$ の出現確率 $P(\mathbf{r}|\theta)$ は

$$P(\mathbf{r}|\theta) = \textstyle\prod_{k=1}^{K} \theta(r_k) \tag{1}$$

と計算される．また，$\sigma(r, \mathbf{r})$ を適用規則列 $\mathbf{r}$ 中に 規則 r が出現する数とすると，

$$P(\mathbf{r}|\theta) = \textstyle\prod_{r \in R} \theta(r)^{\sigma(r, \mathbf{r})} \tag{2}$$

と書くこともできる．$S \overset{*}{\Rightarrow} \mathbf{w}$ を実現する適用規則列すべてから成る集合を $\psi(\mathbf{w})$ とおく．文 $\mathbf{w}$ は適用規則列 $\mathbf{r}$ から一意に定まるので，文と適用規則列の同時分布 $P(\mathbf{w}, \mathbf{r}|\theta)$ に関し

$$P(\mathbf{w}, \mathbf{r}|\theta) = \begin{cases} P(\mathbf{r}|\theta) & \text{if } \mathbf{r} \in \psi(\mathbf{w}) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

が成り立つ．式 3 より開始記号 S から $\mathbf{w}$ が導出される確率 $P(\mathbf{w}|\theta)$ は次のように求まる：

$$P(\mathbf{w}) = \sum_{\text{all } \mathbf{r}} P(\mathbf{w}, \mathbf{r}|\theta) = \sum_{\mathbf{r} \in \psi(\mathbf{w})} P(\mathbf{r}|\theta) \ . \tag{4}$$

パラメタ θ が文脈より明らかなときは $P(\mathbf{r}, \ldots|\theta)$, $P(\mathbf{w}, \ldots|\theta)$ を各々$P(\mathbf{r}, \ldots)$, $P(\mathbf{w}, \ldots)$ と書く．先述した規則適用の独立性に加え，以降で考える PCFG $G(\theta)$ は次を満たすとする．

- $G(\theta)$ は整合的である．すなわち $\sum_{\mathbf{w} \in L_G} P(\mathbf{w}|\theta) = 1$ が成り立つ．
- 右辺が ε である規則（ε 規則）や $A \overset{+}{\Rightarrow} A$ となるような $A \in V_{\mathrm{n}}$ が存在しない．

Chi と Zeman は，2 番目の条件を満たす文法構造 G と有限長の文から成る括弧なしコーパス $\mathcal{C}$ が与えられたとき，I-O アルゴリズムで得られる訓練パラメタ θ^* の下での PCFG $G(\theta^*)$ が整合的であることを示した (Chi and Geman 1998).

2.2　コーパス・構文木

平文 $\mathbf{w} = w_1 w_2 \cdots w_n \in L_G$ に対して個々の w_j は単語である ($n > 0$, $j = 1 \ldots n$)．$\mathbf{w}$ に対して単語位置 $d = 0 \ldots n$ を与える．$0 \leq d \leq d' \leq n$ について d と d' の間にある部分単語列 $w_{d+1} \cdots w_{d'}$ を $\mathbf{w}_{d,d'}$ と書く（$\mathbf{w} = \mathbf{w}_{0,n}$ である）．また，(部分) 単語列 $w_d \cdots w_{d'}$ をリスト $\langle w_d, \ldots, w_{d'} \rangle$ で表すことがある．$\mathbf{w} \in L_G$ に対して，$\mathbf{w}$ の構文木は $S \overset{*}{\Rightarrow} \mathbf{w}$ の導出過程を木構造で表現したものである．我々は導出戦略を最左導出に固定しているので，$S \overset{*}{\Rightarrow} \mathbf{w}$ の適用規則列 $\mathbf{r}$ から $\mathbf{w}$ の構文木 t が一意に決まり，逆も真である．従って $P(t) = P(\mathbf{r})$ が成り立つ．先に我々は対象とする PCFG が ε 規則をもたないこと，$A \overset{+}{\Rightarrow} A$ という導出が起こらないと仮定した．この仮定より，$\mathbf{w}$ の構文木 t の部分構文木（以下，部分木）t' はその根ノードの非終端記号 A と葉ノードを構成する部分単語列の開始／終了位置の対 (d, d') によって一意に定まるので，以降，我々は部分木 t' をラベル $A(d, d')$ で参照する．$\mathbf{w}$ の構文木 t は，葉ノードを除く t の部分木ラベルから成る集合 $\mathcal{L}(t)$ （t のラベル集合と呼ぶ） と同一視できる．また，(d, d') は一つの括弧づけに相当する．$\mathcal{B}(t) \overset{\text{def}}{=} \{(d, d') \mid A(d, d') \in \mathcal{L}(t)\}$ と定め，t の括弧集合と呼ぶ．

また，$A \to \rho_1 \rho_2 \cdots \rho_M$ の展開によって得られた $\rho_1, \rho_2, \ldots, \rho_M$ を根とする部分木 $\rho_m(d_{m-1}, d_m)$ を考える（図 1）．このとき $A(d_0, d_M)@\rho_m(d_{m-1}, d_m)$ なる部分木に関する半順序関係 @ を導入する ($m = 1 \ldots M$)．この関係を以降では「$A(d_0, d_M)$ は $\rho_m(d_{m-1}, d_m)$ の親である」，また逆に「$\rho_m(d_{m-1}, d_m)$ は $A(d_0, d_M)$ の子である」などということがある．そして親 $A(d_0, d_M)$ とその子をすべてまとめて

$$A(d_0, d_M)@\rho_1(d_0, d_1)\rho_2(d_1, d_2) \cdots \rho_M(d_{M-1}, d_M) \tag{5}$$

と表し，これを「部分木の親子対」と呼ぶことにする．構文木 t 中に出現する部分木の親子対をすべて集めた集合を $\mathcal{T}(t)$ で表す．構文木 t に対応する適用規則列 $\mathbf{r}$ に対して $\mathcal{L}(\mathbf{r})$, $\mathcal{B}(\mathbf{r})$, $\mathcal{T}(\mathbf{r})$ をそれぞれ $\mathcal{L}(t)$, $\mathcal{B}(t)$, $\mathcal{T}(t)$ と同一視する．

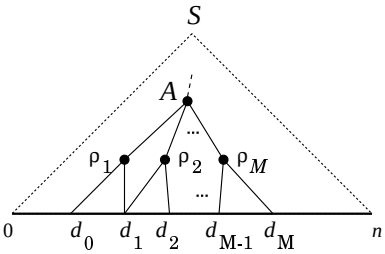

図 1　部分木の親子対.

PCFG の訓練用のコーパスとして我々は (1) 構造つきコーパス (labeled corpus), (2) 完全括弧つきコーパス (fully bracketed corpus), (3) 部分括弧つきコーパス (partially bracketed corpus), (4) 括弧なしコーパス (unbracketed corpus) の 4 つを考える. 我々は訓練法として最尤推定法 (maximum likelihood estimation) を考えており, N 文を含むコーパス $\mathcal{C} \stackrel{\text{def}}{=} \langle c_1, c_2, \ldots, c_N \rangle$ は PCFG $G(\theta)$ に基づく独立な N 回のサンプリング導出の結果であると仮定する. $\mathbf{w}_\ell$, $\mathbf{r}_\ell$ を ℓ 回目のサンプリングで得られた平文および適用規則列とすると ($\ell = 1 \ldots N$), $\mathcal{C}$ が構造つきコーパスのとき $c_\ell = \langle \mathbf{w}_\ell, \mathcal{L}(\mathbf{r}_\ell) \rangle$, 完全括弧つきコーパスのとき $c_\ell = \langle \mathbf{w}_\ell, \mathcal{B}(\mathbf{r}_\ell) \rangle$, 部分括弧つきコーパスのとき $c_\ell = \langle \mathbf{w}_\ell, \mathcal{B}_\ell \rangle$, 括弧なしコーパスのとき $c_\ell = \mathbf{w}_\ell$ となる ($\mathbf{w}_\ell \in L_G$, $\mathbf{r} \in \psi(\mathbf{w}_\ell)$, $\mathcal{B}_\ell \subseteq \mathcal{B}(\mathbf{r}_\ell)$). $\mathbf{w}_\ell$ の部分単語列を $\mathbf{w}_{d,d'}^{(\ell)}$ とおき ($0 \leq d < d' \leq n_\ell$), $\mathbf{w}_\ell$ の d 番目の単語を $w_d^{(\ell)}$ とおく. ただし $n_\ell \stackrel{\text{def}}{=} |\mathbf{w}_\ell|$ である.

2.3　CYK パーザ

CYK パーザは Chomsky 標準形である CFG に適用可能なパーザである. 我々は括弧なしコーパス $\mathcal{C}$ 中の文 $\mathbf{w}_\ell$ に対して $n_\ell \times n_\ell$ の三角行列 $T^{(\ell)}$ を用意する[3]($n_\ell = |\mathbf{w}_\ell|$). d 行 d' 列の要素 $T_{d,d'}$ には部分単語列 $\mathbf{w}_{d,d'}$ に対する解析結果が格納される. CYK パーザを実現する手続き *CYK-Parser* を図 2 に示す. 対角要素から順に部分木を組み上げ（行 4–9）, T_{0,n_ℓ} に $S(0,n_\ell)@\cdot$ が含まれていたら解析が成功したものとし, 含まれていなかったら失敗したものとする（行 10）. 解析が成功したら T_{0,n_ℓ} に含まれる $S(0,n_\ell)@\cdot$ から順に部分木の親子対を辿って構文木が取り出される. 図 3 に示した CFG $G1$ において文 $\mathbf{w} = \langle$ 急いで, 走る, 一郎, を, 見た $\rangle$ に対する三角行列を図 4 に示す. 図 4 の ○ 印のついた部分木の親子から図 5 の構文木 $t1$ が取り出され, ● 印のついた部分木の親子から $t2$ が取り出される.

3 通常, 三角行列の各行と各列に振られる番号は（単語位置ではなく）単語そのものに振られた番号であるが, 本論文では他の記法と整合をとるために行番号を 1 つずつ減らす.

procedure *CYK-Parser*($\mathcal{C}$) **begin**
  for $\ell := 1$ **to** N **do begin**
    Prepare $(n_\ell \times n_\ell)$ triangular matrix $T^{(\ell)}$;
    for $d := 0$ **to** $n_\ell - 1$ **do**　　/* 三角行列の対角要素から始める */
      $T^{(\ell)}_{d,d+1} := \{A(d,d+1)@w^{(\ell)}_{d+1}(d,d+1) \mid (A \to w_{d+1}) \in R\}$;
    for $k := 2$ **to** n_ℓ **do**　　/* 三角行列の非対角要素について計算 */
      for $d := 0$ **to** $n_\ell - k$ **do**
        $T^{(\ell)}_{d,d+k} := \bigcup_{k'=1}^{k-1} \{A(d,d+k)@B(d,d+k')C(d+k',d+k) \mid (A \to BC) \in R,$
          $(B(d,d+k')@\cdot) \in T^{(\ell)}_{d,d+k'}, (C(d+k',d+k)@\cdot) \in T^{(\ell)}_{d+k',d+k}\}$;
    if $(S(0,n_\ell)@\cdot) \in T^{(\ell)}_{0,n_\ell}$ **then** accept **else** reject
  end
end.

図 2　CYK パーザ.

$G1$:

(1)	S → PP V	(6)	NP → V N	(10)	N → 一郎
(2)	S → ADV VP	(7)	PP → NP P	(11)	P → を
(3)	VP → PP V	(8)	PP → N P	(12)	V → 走る
(4)	VP → ADV V	(9)	ADV → 急いで	(13)	V → 見た
(5)	NP → VP N				

図 3　文脈自由文法の例 $G1$.

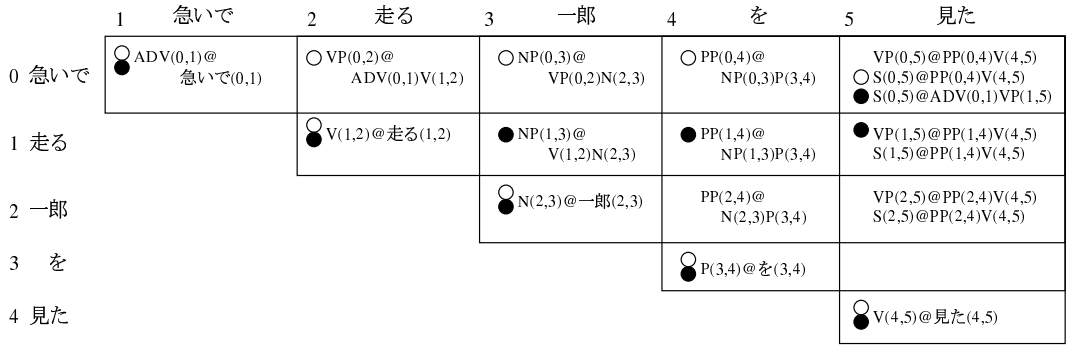

図 4　$G1$ と文 ⟨ 急いで, 走る, 一郎, を, 見た ⟩ に対する三角行列.

2.4 Inside-Outside アルゴリズム

先にも述べたように，我々は PCFG のパラメタをコーパス $\mathcal{C} = \langle c_1, c_2, \ldots, c_N \rangle$ から最尤推定法に基づき訓練することを考えている．$\mathcal{C}$ が構造つきコーパスの場合，相対頻度法で得られる各規則 r の相対出現頻度が最尤推定値となるので，これを r の訓練パラメタ $\theta^*(r)$ とすれば

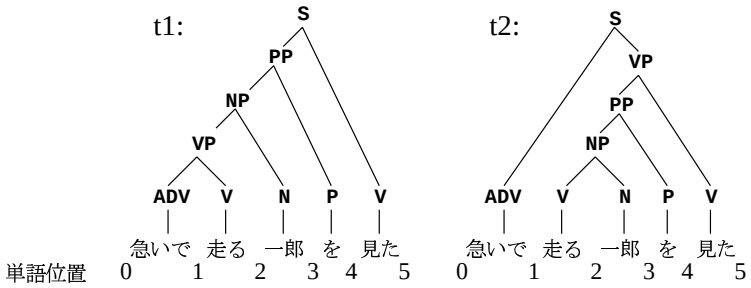

図 5　三角行列から取り出された 2 つの構文木.

よい．しかし，構造つきコーパスの作成コストを考えると，より安価な括弧なしコーパスしか利用できない場合が十分考えられる．括弧なしコーパスでは構文構造が明らかでないため，相対頻度法が適用できず，代わりに I-O アルゴリズムという PCFG に特化された形の EM アルゴリズムが広く知られている．I-O アルゴリズムは，括弧なしコーパス $\mathcal{C} = \langle \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N \rangle$ が与えられたときに尤度 $\prod_{\ell=1}^{N} P(\mathbf{w}_\ell \mid \theta)$ あるいはその対数 $\sum_{\ell=1}^{N} \log P(\mathbf{w}_\ell \mid \theta)$ （対数尤度）を局所的に最大にする θ^* を見つける．つまり I-O アルゴリズムもまた最尤推定法である．

(Lari and Young 1990) をはじめとする多くの文献の記述では，文法構造 G のうち規則集合 R を明示的に与えず，終端記号集合 V_t と非終端記号集合 V_n を与えた場合を考えている．提案手法との対比のため，本節では R を明示的に与えた場合の I-O アルゴリズムを記述する．V_t と V_n のみを与えた場合の I-O アルゴリズムは

$$R_{\max}(V_\mathrm{n}, V_\mathrm{t}) \stackrel{\mathrm{def}}{=} \{A \to BC \mid A, B, C \in V_\mathrm{n}\} \cup \{A \to a \mid A \in V_\mathrm{n}, a \in V_\mathrm{t}\} \tag{6}$$

（以下では $R_{\max}$ と略すことがある）を考え，規則集合を $R = R_{\max}(V_\mathrm{n}, V_\mathrm{t})$ として与えた場合と同一である．ただし，いずれの場合でも R は Chomsky 標準形でなければならない．(Lari and Young 1990) では規則集合 R を含めた学習を目的に I-O アルゴリズムを使用しているが，我々はパラメタ θ の学習に焦点を絞る．

I-O アルゴリズムの中心は内側確率 $P(A \stackrel{*}{\Rightarrow} \mathbf{w}_{d,d'}^{(\ell)})$ と外側確率 $P(S \stackrel{*}{\Rightarrow} \mathbf{w}_{0,d}^{(\ell)} A \mathbf{w}_{d',n_\ell}^{(\ell)})$ という 2 つの確率値の計算である ($\ell = 1 \ldots N$, $A \in V_\mathrm{n}$, $0 \le d < d' \le n_\ell$)．各確率値を配列変数 $\beta_{d,d'}^{(\ell)}[A]$, $\alpha_{d,d'}^{(\ell)}[A]$ に格納する．これらの配列変数は CYK アルゴリズムで用いた三角行列 $T_{d,d'}$ 中に設けられているものとする．$\mathbf{w}_{0,n_\ell}^{(\ell)} = \mathbf{w}_\ell$ より $\beta_{0,n_\ell}^{(\ell)}[S]$ に文 $\mathbf{w}_\ell$ の生起確率 $P(S \stackrel{*}{\Rightarrow} \mathbf{w}_\ell)$ が格納される点に注意する．

内側確率と外側確率を計算する手続き *Get-Beta*, *Get-Alpha* を図 6 に示す．記述を簡単にするため，配列変数 $\alpha_{d,d'}^{(\ell)}[\cdot]$ および $\beta_{d,d'}^{(\ell)}[\cdot]$ は手続きが呼び出される度に 0 に初期化されるものとする．*Get-Beta* は CYK パーザにおいて部分木を組み上げるのと同じように，三角行列の対角要素から出発し，右上隅 $\beta_{0,n_\ell}^{(\ell)}[\cdot]$ に至るまで段階的に内側確率を計算していく．また，逆に

Get-Alpha では右上隅 $\alpha^{(\ell)}_{0,n_\ell}[\cdot]$ から対角要素に向かって外側確率を計算する．このように内側・外側確率は動的計画法 (dynamic programming) に基づき，一方向に従って計算が進められる．内側・外側確率を計算し終ったら，コーパス $\mathcal{C}$ が与えられた下での規則 $A\to BC$, $A\to a$ の適用回数の条件つき期待値（以下，期待適用回数という）が次のように計算される:

$$\eta[A\to BC] := \sum_{\ell=1}^{N}\frac{1}{\beta^{(\ell)}_{0,n_\ell}[S]}\sum_{k=2}^{n_\ell}\sum_{d=0}^{n_\ell-k}\sum_{k'=1}^{k-1}\theta(A\to BC)\alpha^{(\ell)}_{d,d+k}[A]\beta^{(\ell)}_{d,d+k'}[B]\beta^{(\ell)}_{d+k',d+k}[C], \tag{7}$$

$$\eta[A\to a] := \sum_{\ell=1}^{N}\frac{1}{\beta^{(\ell)}_{0,n_\ell}[S]}\sum_{d=0}^{n_\ell-1}\theta(A\to a)\alpha^{(\ell)}_{d,d+1}[A]. \tag{8}$$

更に，上で計算された期待値からパラメタ $\theta(A\to\zeta)$ が更新（再推定）される:

$$\theta(A\to\zeta) := \eta[A\to\zeta]\Big/\textstyle\sum_{\zeta':(A\to\zeta')\in R}\eta[A\to\zeta']. \tag{9}$$

I-O アルゴリズムでは，まず θ を適当な値に初期化し，次いで手続き *Get-Beta*, *Get-Alpha* および式 7, 8, 9 によって θ を更新する．そして，このように更新を繰り返すと対数尤度 $\sum_{\ell=1}^{N}\log P(\mathbf{w}_\ell)=\sum_{\ell=1}^{N}\log\beta^{(\ell)}_{0,n_\ell}[S]$ が単調増加しながら最終的には収束する．収束したら，そ

```
1:  procedure Get-Beta() begin
2:    for ℓ := 1 to N do begin
3:      for d := 0 to n_ℓ − 1 do    /* 三角行列の対角要素について計算 */
4:        foreach A such that (A→w^(ℓ)_{d+1}) ∈ R do
5:          β^(ℓ)_{d,d+1}[A] := θ(A→w^(ℓ)_{d+1});
6:      for k := 2 to n_ℓ do    /* 三角行列の非対角要素について計算 */
7:        for d := 0 to n_ℓ − k do
8:          foreach A ∈ V_n do
9:            β^(ℓ)_{d,d+k}[A] := Σ_{B,C:(A→BC)∈R} θ(A→BC) Σ_{k'=1}^{k−1} β^(ℓ)_{d,d+k'}[B] β^(ℓ)_{d+k',d+k}[C]
10:   end
11: end.
```

```
1:  procedure Get-Alpha() begin
2:    for ℓ := 1 to N do begin
3:      α^(ℓ)_{0,n_ℓ}[S] := 1;    /* 右上隅の S については特別に 1 に初期化 */
4:      for k := n_ℓ downto 2 do
5:        for d := 0 to n_ℓ − k do
6:          foreach B ∈ V_n do
7:            α^(ℓ)_{d,d+k}[B] := Σ_{A,X:(A→BX)∈R} θ(A→BX) Σ_{k'=k+1}^{n_ℓ−d} α^(ℓ)_{d,d+k'}[A] β^(ℓ)_{d+k,d+k'}[X]
8:                              + Σ_{A',Y:(A'→YB)∈R} θ(A'→YB) Σ_{k'=1}^{d} α^(ℓ)_{d−k',d+k}[A'] β^(ℓ)_{d−k',d}[Y]
9:    end
10: end.
```

図 6　（上）内側確率の計算ルーチン *Get-Beta*,（下）外側確率の計算ルーチン *Get-Alpha*.

のときのパラメタの値を最終的な推定値として I-O アルゴリズムは終了する．

ここで，I-O アルゴリズムの計算量を考える．収束までのパラメタ更新回数は初期値に依存するため，事前には分からない．従って 1 回のパラメタ更新に必要な計算量を I-O アルゴリズムの計算量とする．非終端記号集合 V_{n}, 終端記号集合 V_{t} を固定した場合の最悪計算量を測る場合には $R = R_{\max}(V_{\mathrm{n}}, V_{\mathrm{t}})$ の場合を考えればよい．訓練コーパス $\mathcal{C}$ に対して最長の文の長さを L とする．手続き *Get-Beta*, *Get-Beta*（図 6）中の **for**, **foreach** ループと $\sum$ の引数に注目すれば，I-O アルゴリズムの最悪計算量は $O(|V_{\mathrm{n}}|^3 L^3)$ であることが容易に分かる．

2.5 Inside-Outside アルゴリズムに関する考察

アルゴリズム中で最もコストが高いのは，*Get-Beta* 行 9 における内側確率の計算，*Get-Alpha* 行 7–8 における外側確率の計算である．*Get-Beta* 行 9 において図 7 (1) という状況すべてを考慮して内側確率が計算される．一方，*Get-Alpha* の行 7–8 における右辺第 1 項，第 2 項ではそれぞれ図 7 (2), (3) という状況がすべて考慮されている．考えられるすべての状況について計算をすすめるという意味で I-O アルゴリズムの動作は仮説駆動（トップダウン）型パーザの動作と同じである．一般に仮説駆動型は入力文 $\mathbf{w}_\ell$ の情報とは無関係に計算をすすめるために効率が悪いとされている．文法構造が与えられていても I-O アルゴリズムの計算速度が低いのは，仮説駆動型であることが原因であると考えられる．そもそも I-O アルゴリズムは

$$\eta[r] = \sum_{\ell=1}^{N} \sum_{\text{all } \mathbf{r}} P(\mathbf{r}|\mathbf{w}_\ell)\sigma(r,\mathbf{r}) = \sum_{\ell=1}^{N} \frac{1}{P(\mathbf{w}_\ell)} \sum_{\text{all } \mathbf{r}} P(\mathbf{w}_\ell,\mathbf{r})\sigma(r,\mathbf{r}) \tag{10}$$

という規則 r の期待適用回数 $\eta[r]$ の計算を

$$\eta[r] \;=\; \theta(r)\cdot\sum_{\ell=1}^{N}\frac{1}{P(\mathbf{w}_\ell)}\frac{\partial P(\mathbf{w}_\ell)}{\partial\theta(r)} \;=\; \theta(r)\cdot\sum_{\ell=1}^{N}\frac{1}{P(\mathbf{w}_\ell)}\frac{\partial}{\partial\theta(r)}\sum_{\text{all }\mathbf{r}}P(\mathbf{w}_\ell,\mathbf{r}) \tag{11}$$

から得られる手続き *Get-Beta*, *Get-Alpha* および式 7, 8 によって効率化したものである (Lafferty 1993)[4]．ただし，節 2.1で定めたように $\sigma(r,\mathbf{r})$ は規則列 $\mathbf{r}$ に出現する規則 r の数である．式 11 で，$r = (A \to BC)$ とおいたとき，I-O アルゴリズムでは $\frac{\partial}{\partial\theta(A\to BC)}\sum_{\text{all }\mathbf{r}} P(\mathbf{w}_\ell,\mathbf{r})$ を次のように計算する（添字の $\cdot_\ell$, $\cdot^{(\ell)}$ は省略）．

$$\begin{aligned}
&\frac{\partial}{\partial\theta(A\to BC)}\sum_{\text{all }\mathbf{r}}P(\mathbf{w},\mathbf{r}) \\
&\quad= \frac{\partial}{\partial\theta(A\to BC)}\sum_{\text{all }\mathbf{r}\text{ s.t. }A\to BC\text{ appears in }\mathbf{r}} P(\mathbf{w},\mathbf{r}) \\
&\quad= \frac{\partial}{\partial\theta(A\to BC)}\sum_{d,k,k'}\sum_{\substack{\text{all }\mathbf{r}\text{ s.t. }A\to BC\text{ appears in }\mathbf{r}\\ \text{with the position }(d,d+k',d+k)}} P(\mathbf{w},\mathbf{r})
\end{aligned} \tag{12}$$

4 (Lafferty 1993) ではコーパスサイズ $N = 1$ の場合が説明されている．

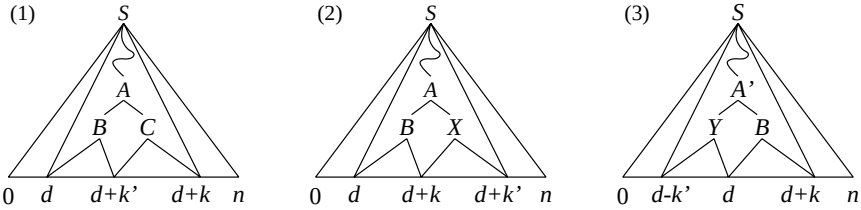

図 7　内側確率，外側確率の計算において想定している状況（$n = n_\ell$ とおいている）．

$$
\begin{aligned}
&= \frac{\partial}{\partial\theta(A \to BC)} \sum_{d,k,k'} P(S \overset{*}{\Rightarrow} \mathbf{w}_{0,d} A \mathbf{w}_{d+k,n}) \theta(A \to BC) \cdot \\
&\qquad\qquad P(B \overset{*}{\Rightarrow} \mathbf{w}_{d,d+k'}) P(C \overset{*}{\Rightarrow} \mathbf{w}_{d+k',d+k}) \\
&= \sum_{d,k,k'} P(S \overset{*}{\Rightarrow} \mathbf{w}_{0,d} A \mathbf{w}_{d+k,n}) P(B \overset{*}{\Rightarrow} \mathbf{w}_{d,d+k'}) P(C \overset{*}{\Rightarrow} \mathbf{w}_{d+k',d+k}) \qquad (13)
\end{aligned}
$$

式 12 の変形は入力文 $\mathbf{w}$ や実際の構文木 $t \in \psi(\mathbf{w})$ とは無関係に行なわれており，I-O アルゴリズムが仮説駆動型であるというのはこの点に由来する．

それに対し，式 3 より式 10 を下の式 14 に変形し，構文木情報 ψ を直接利用する方法を考える．ψ はパーザを利用することによって事前に獲得しておく．また，式 14 は Fujisaki らの計算方法 (Fujisaki et al. 1989) に他ならない．

$$
\eta[r] = \sum_{\ell=1}^{N} \frac{1}{P(\mathbf{w}_\ell)} \sum_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} P(\mathbf{r}) \sigma(r, \mathbf{r}) \qquad (14)
$$

式 14 を用いれば I-O アルゴリズムのように ψ と無関係な部分を計算することはなくなる．ただし，一般に $|\psi(\mathbf{w})|$ は文長 $|\mathbf{w}|$ に対して指数オーダになってしまうため，これをそのまま計算するのは現実的ではない．提案手法では I-O アルゴリズムのように再計算を防ぐ仕組みを取り入れ，パーザのもつ WFST を利用して式 14 を効率的に計算する．従って，提案手法を Fujisaki らの方法と I-O アルゴリズム双方の長所を取り入れた方法と見ることもできる．

3　提案手法

提案手法の概要を図 8 に示す．入力として確率文脈文法 $G(\theta)$ の文法構造 $\langle V_{\mathrm{n}}, V_{\mathrm{t}}, R, S \rangle$ と括弧なしコーパス $\mathcal{C}$ が与えられるものとする．そして訓練パラメタ θ を出力として返す．提案手法において，我々は全体の訓練過程を構文解析と EM 学習に分離する．はじめに我々はパーザで $\mathcal{C}$ 中の各文 $\mathbf{w}_\ell$ をすべて解析する．すると $\psi(\mathbf{w}_\ell)$ を細切れにした，しかし $\psi(\mathbf{w}_\ell)$ と等価な構文情報 $\langle O_\ell, \tilde{\psi}_\ell \rangle$ がパーザの WFST に格納されているので，これらを抽出する．$\langle O_\ell, \tilde{\psi}_\ell \rangle$ を表現するデータ構造を支持グラフと呼ぶ．次に，支持グラフに基づき gEM アルゴリズムを動作させ θ を得る．図 3 の CFG $G1$ と文 $\mathbf{w}_\ell = \langle$ 急いで, 走る, 一郎, を, 見た $\rangle$ の例を考えると，

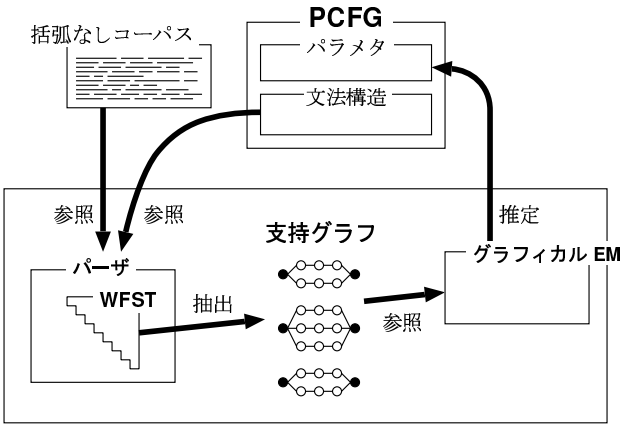

図 8　提案手法の概要.

支持グラフは図 4 において ○ 印と ● 印がついた部分木の親子から得られる．この例から分かるように，文法によっては gEM アルゴリズムで参照する支持グラフは三角行列全体に比べて非常に小さくなる可能性があり，その場合は三角行列全体を走査しなければならない I-O アルゴリズムに比べ大幅な速度向上が得られる（提案手法の**特長 2**）．

3.1　準備

提案手法を記述する前に形式化を行なう．$\ell = 1 \ldots N$ について以下をおこなう．まず，$\mathcal{T}_\ell \stackrel{\text{def}}{=} \bigcup_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} \mathcal{T}(\mathbf{r})$, $V_\ell \stackrel{\text{def}}{=} \bigcup_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} \mathcal{L}(\mathbf{r}) = \{\tau \mid \tau @ \cdot \in \mathcal{T}_\ell\}$ と定める[5]．そして，V_ℓ の要素を $\tau_k @ \tau_{k_1} \tau_{k_2} \cdots \tau_{k_M} \in \mathcal{T}_\ell \Rightarrow k < k_1, k_2, \ldots, k_M$ を満たすように並べた $\langle \tau_1, \tau_2, \ldots, \tau_{|V_\ell|} \rangle$ を O_ℓ とする．また，次を導入する．

$$\tilde{\psi}_\ell(A(d,d')) \stackrel{\text{def}}{=} \left\{ E \;\middle|\; \begin{array}{l} A(d,d') @ \rho_1(d_0,d_1)\rho_2(d_1,d_2)\cdots\rho_M(d_{M-1},d_M) \in \mathcal{T}_\ell, \\ E = \{\rho_m(d_{m-1},d_m) \mid m = 1 \ldots M\} \\ \qquad\qquad \cup \{A \to \rho_1\rho_2\cdots\rho_M\}, \quad d = d_0,\ d' = d_M \end{array} \right\} \tag{15}$$

$\mathcal{T}_\ell$ はコーパス中の文 $\mathbf{w}_\ell$ の構文木のいずれかに現れる部分木の親子対の集合である．同様に V_ℓ は $\mathbf{w}_\ell$ の構文木のいずれかに現れる部分木ラベルの集合である．O_ℓ は V_ℓ の要素を $\mathcal{T}_\ell$ 中の半順序関係（親子関係）@ を満たすように順序づけたものである．O_ℓ の第一要素 τ_1 は必ず $S(0, n_\ell)$ になる．$\tilde{\psi}_\ell$ は部分木と規則の論理的な関係を表現する．例えば，

$$\tilde{\psi}_\ell(A(d,d')) = \{\ \{A \to B_1C_1,\ B_1(d,d_1''),\ C_1(d_1'',d')\},\ \{A \to B_2C_2,\ B_2(d,d_2''),\ C_2(d_2'',d')\}\ \}$$

に対しては，「$\mathbf{w}_\ell$ に対して部分木 $A(d,d')$ を作るためには，規則 $A \to B_1C_1$ を適用し，部分木 $B_1(d,d_1'')$ と部分木 $C_1(d_1'',d')$ を作る，もしくは 規則 $A \to B_2C_2$ を適用し，部分木 $B_2(d,d_2'')$

5 $\mathcal{L}(\mathbf{r})$ および $\mathcal{T}(\mathbf{r})$ については節 2.2 で定めたとおりである．

と部分木 $C_2(d_2'', d')$ を作る，のいずれかである（他の場合はあり得ない）」と解釈する．O_ℓ と $\tilde{\psi}_\ell$ は次節で説明する支持グラフを構成する．

例として，図 3 の CFG $G1$ と $\mathbf{w}_\ell = \langle$ 急いで, 走る, 一郎, を, 見た $\rangle$ に対して図 5 の 2 つの構文木 $t1, t2$ を考える．各々に対応する適用規則列を $\mathbf{r}_1, \mathbf{r}_2$ とおくと，$\psi(\mathbf{w}_\ell) = \{\mathbf{r}_1, \mathbf{r}_2\}$ である．このとき $\mathcal{T}_\ell$ は

$$
\begin{aligned}
\mathcal{T}_\ell &= \mathcal{T}(\mathbf{r}_1) \cup \mathcal{T}(\mathbf{r}_2) \\
&= \{\, \mathtt{S}(0,5)@\mathtt{PP}(0,4)\mathtt{V}(4,5),\ \mathtt{PP}(0,4)@\mathtt{NP}(0,3)\mathtt{P}(3,4),\ \mathtt{NP}(0,3)@\mathtt{VP}(0,2)\mathtt{N}(2,3), \\
&\qquad \mathtt{VP}(0,2)@\mathtt{ADV}(0,1)\mathtt{V}(1,2),\ \mathtt{V}(4,5)@見た\,(4,5),\ \mathtt{P}(3,4)@を\,(3,4),\ \mathtt{N}(2,3)@一郎\,(2,3), \\
&\qquad \mathtt{V}(1,2)@走る\,(1,2),\ \mathtt{ADV}(0,1)@急いで\,(0,1)\,\} \\
&\quad \cup \{\, \mathtt{S}(0,5)@\mathtt{ADV}(0,1)\mathtt{VP}(1,5),\ \mathtt{ADV}(0,1)@急いで\,(0,1),\ \mathtt{VP}(1,5)@\mathtt{PP}(1,4)\mathtt{V}(4,5), \\
&\qquad \mathtt{PP}(1,4)@\mathtt{NP}(1,3)\mathtt{P}(3,4),\ \mathtt{NP}(1,3)@\mathtt{V}(1,2)\mathtt{N}(2,3),\ \mathtt{V}(1,2)@走る\,(1,2), \\
&\qquad \mathtt{N}(2,3)@一郎\,(2,3),\ \mathtt{P}(3,4)@を\,(3,4),\ \mathtt{V}(4,5)@見た\,(4,5)\,\}
\end{aligned}
$$

となる．また，O_ℓ は一意には決まらないが，どの場合でも第一要素は必ず $\mathtt{S}(0,5)$ になる点に注意する．例えば下のような O_ℓ が考えられる．

$$
\begin{aligned}
O_\ell = \langle &\mathtt{S}(0,5), \mathtt{VP}(1,5), \mathtt{PP}(1,4), \mathtt{NP}(1,3), \mathtt{V}(4,5), \mathtt{PP}(0,4), \mathtt{P}(3,4), \\
&\mathtt{NP}(0,3), \mathtt{N}(2,3), \mathtt{VP}(0,2), \mathtt{V}(1,2), \mathtt{ADV}(0,1)\rangle
\end{aligned}
$$

また $\tilde{\psi}_\ell$ を O_ℓ の順に示す．

$$
\begin{aligned}
\tilde{\psi}_\ell(\mathtt{S}(0,5)) &= \{\, \{\mathtt{S}\to\mathtt{PP\ V},\ \mathtt{PP}(0,4),\ \mathtt{V}(4,5)\}, \\
&\qquad \{\mathtt{S}\to\mathtt{ADV\ VP},\ \mathtt{ADV}(0,1),\ \mathtt{VP}(1,5)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{VP}(1,5)) &= \{\, \{\mathtt{VP}\to\mathtt{PP\ V},\ \mathtt{PP}(1,4),\ \mathtt{V}(4,5)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{PP}(1,4)) &= \{\, \{\mathtt{PP}\to\mathtt{NP\ P},\ \mathtt{NP}(1,3),\ \mathtt{P}(3,4)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{NP}(1,3)) &= \{\, \{\mathtt{NP}\to\mathtt{V\ N},\ \mathtt{NP}(1,2),\ \mathtt{N}(2,3)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{V}(4,5)) &= \{\, \{\mathtt{V}\to見た\,\}\,\} \\
\tilde{\psi}_\ell(\mathtt{PP}(0,4)) &= \{\, \{\mathtt{PP}\to\mathtt{NP\ P},\ \mathtt{NP}(0,3),\ \mathtt{P}(3,4)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{P}(3,4)) &= \{\, \{\mathtt{P}\to を\,\}\,\}
\end{aligned}
\qquad
\begin{aligned}
\tilde{\psi}_\ell(\mathtt{NP}(0,3)) &= \{\, \{\mathtt{NP}\to\mathtt{VP\ N}, \\
&\qquad \mathtt{VP}(0,2),\ \mathtt{N}(2,3)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{N}(2,3)) &= \{\, \{\mathtt{N}\to一郎\,\}\,\} \\
\tilde{\psi}_\ell(\mathtt{VP}(0,2)) &= \{\, \{\mathtt{VP}\to\mathtt{ADV\ V}, \\
&\qquad \mathtt{ADV}(0,1),\ \mathtt{V}(1,2)\}\,\} \\
\tilde{\psi}_\ell(\mathtt{V}(1,2)) &= \{\, \{\mathtt{V}\to走る\,\}\,\} \\
\tilde{\psi}_\ell(\mathtt{ADV}(0,1)) &= \{\, \{\mathtt{ADV}\to急いで\,\}\,\}
\end{aligned}
$$

3.2　支持グラフ

$\langle O_\ell, \tilde{\psi}_\ell\rangle$ という組を支持グラフ Δ_ℓ というデータ構造で捉えると gEM アルゴリズムが理解しやすくなる．「グラフィカル EM」の名もここに由来する．まず，前節で示した O_ℓ, $\tilde{\psi}_\ell$ の例に対応する支持グラフを図 9 (a) に示す．支持グラフ Δ_ℓ は再帰遷移ネットワーク（recursive transition network; 以下 RTN）に似た構造をもつ非循環有向グラフ (DAG) であり，共通の辺をもたない部分グラフ $\tilde{\Delta}_\ell(\tau) \stackrel{\text{def}}{=} \langle \tau, \tilde{\psi}_\ell(\tau)\rangle$ の集まりから成る（ただし $\tau \in O_\ell$）．各 $\tilde{\Delta}_\ell(\tau)$ は「τ の部分支持グラフ」と呼ばれ，$\tau = A(d, d')$ が付与されている．また，$\tilde{\Delta}_\ell(\tau)$ は開始ノード，終了ノードと呼ばれる 2 つの特殊なノードをもち（図 9 では各々 start, end と書かれている），各 $E \in \tilde{\psi}_\ell(\tau)$ に対して開始ノード，E の各要素（規則 $A\to\zeta$ または部分木ラベル $A(d, d')$）が

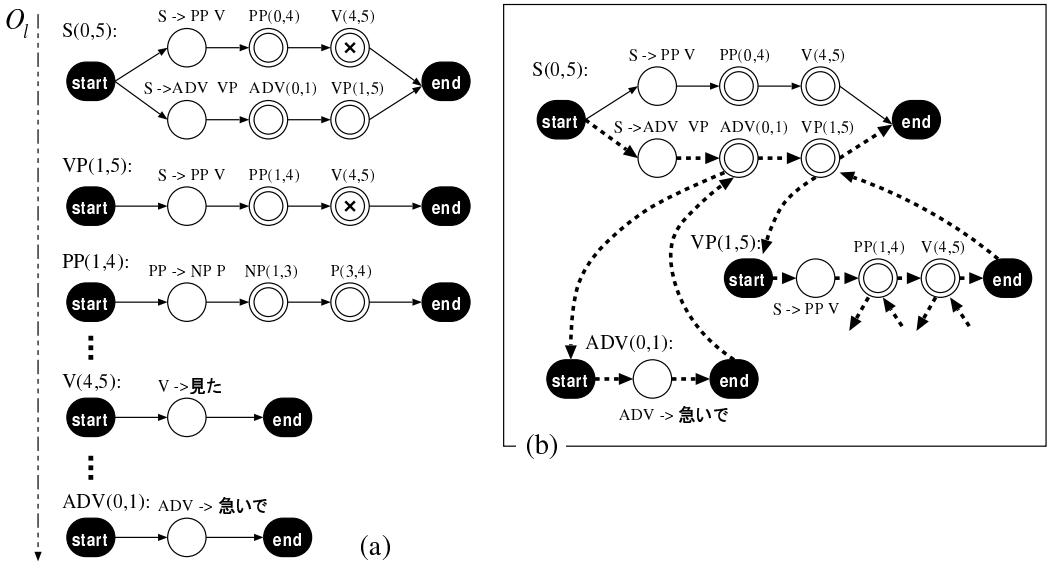

図 9　支持グラフの例.

付与されたノード，終了ノードが一列に連結されている．複数のノードに同じ規則または部分木ラベルが付与されることもある点に注意する．有向辺はすべて開始ノードから終了ノードに向かっている．開始ノードから終了ノードに至るパスを局所パスと呼び，これも E で参照する．局所パスにおいて，規則 $A \rightarrow \zeta$ が付与されたノードを基本ノード，部分木ラベル $A(d, d')$ が付与されたノードを中間ノードと呼び，各々図 9 のように ○ と ◎ で表す．支持グラフは次の特徴をもつ．

(1)　支持グラフ Δ_ℓ に対して RTN のように再帰的な巡回を行なうことができる．

(2)　複数の巡回パスの一部が共有される．

(3)　部分支持グラフ $\tilde{\Delta}_\ell(\tau) = \langle \tau, \tilde{\psi}(\tau) \rangle$ の $E \in \tilde{\psi}_\ell(\tau)$ に対して，どの $\tau' = A(d, d') \in E$ についても $\tau @ \tau'$ が成り立つ．

(4)　一つの局所パス中に存在する基本ノードと中間ノードの数に制限がない．

1 つ目の特徴である再帰的な巡回は次のようにして行なわれる．$S(0, n_\ell)$ の開始ノードから出発し，辺に沿って各ノードを訪問していくが，途中に中間ノード $\tau = A(d, d')$ があったら，τ が付与された部分支持グラフ τ の開始ノードにジャンプする．そして終了ノードに至ったらジャンプ元のノードに戻る．これを再帰的に繰り返し，$S(0, n_\ell)$ の終了ノードに至ったら一回の巡回を終了する．分岐がある場合はその中のどれかを選ぶ．このような巡回の途中で中間ノードに付与される部分木ラベルを集めると $\mathbf{w}_\ell$ の構文木いずれか一つのラベル集合が得られる．また，局所パス中のノードの順序を図 9 のようにして，巡回中に基本ノードに付与されている規則を

順に集めると $\mathbf{w}_\ell$ の最左導出における適用規則列 $\mathbf{r} \in \psi(\mathbf{w}_\ell)$ が一つ得られる．再帰的巡回を全通り行なえば $\psi(\mathbf{w}_\ell)$ 中の適用規則列をすべて見つけることができる．この考えは後に記述する gEM アルゴリズムの正当性を示すときに用いる（付録 A）．図 9 (b) に再帰的巡回の例を示す．

2 つ目の特徴が得られるのは，ある再帰的巡回において，同じ部分木ラベル $\tau = A(d, d')$ が付与されたノードでは同じ部分支持グラフ $\tilde{\Delta}_\ell(\tau)$ にジャンプするためである．このような共有構造により支持グラフのサイズが圧縮され，我々は gEM アルゴリズムを支持グラフの上で動作させることによって効率的な確率計算を実現する．例えば，図 9 (a) において $\mathtt{V}(4,5)$ が付与されたノード（× 印）では同じ部分支持グラフ $\tilde{\Delta}_\ell(\mathtt{V}(4,5))$ にジャンプする．

3 つ目の特徴は，ε 規則およびサイクル $A \overset{+}{\Rightarrow} A$ が存在しないという仮定と，O_ℓ, $\tilde{\psi}_\ell$ の定義から明らかであり，「$\tau @ \tau'$ であるとき，τ' の部分支持グラフ $\tilde{\Delta}_\ell(\tau')$ 中のノードは τ を参照しない」と言い替えることもできる．この事実に基づき，I-O アルゴリズムの内側・外側確率計算における動的計画法（節 2.4）の考えを一般化したものが gEM アルゴリズムに導入されている．また，4 つ目の特徴は支持グラフの構造の一般性を示しているが，gEM アルゴリズムはこの一般性を保持するように記述される．

3.3　支持グラフの獲得

次に，支持グラフ $\langle O_\ell, \tilde{\psi}_\ell \rangle$ をパーザがもつ WFST から効率的に抽出する方法を説明する．O_ℓ は V_ℓ の要素を $\mathcal{T}_\ell$ における半順序関係 @ を満たすように全順序に並べたものである．一般に，半順序関係の全順序関係への変換はトポロジカルソーティングによって実現される．従って，我々はトポロジカルソーティングの考えに基づき O_ℓ を獲得する．また，ソーティングの途中で $\tilde{\psi}_\ell$ が計算できる．以上を実現する支持グラフ抽出ルーチン *Extract-CYK* を図 10（上）に示す．ただし，そのサブルーチンは利用するパーザの WFST の形式に特化したものを用意する．図 10（下）に CYK 用サブルーチン *Visit-CYK* を示す．

我々は大域的にスタック[6] U とフラグ $Visited[\cdot]$ を用意し，再帰的手続き *Visit-CYK* で三角行列（CYK の WFST）の右上隅から部分木 $A(d, d')$ を次々に訪問する（*Extract-CYK* 行 5）．そして訪問が終ったら，部分木のラベルをスタック U に積む（*Visit-CYK* 行 10）．また，訪問の途中で $\tilde{\psi}_\ell$ を記録していく（*Visit-CYK* 行 3, 6）．フラグ $Visited[\cdot]$ に訪問したことを記録し，一度訪問した部分木には行かない（*Visit-CYK* 行 2, 7–8）．最後にスタック U に積んであった部分木ラベルを順に取り出せば（*Extract-CYK* 行 6–7），それが O_ℓ になっている．

GLR パーザの WFST である共有圧縮統語森は $\mathcal{T}_\ell$ を木（森）構造で捉えたものと見ることができる．GLR パーザは文法構造に Chomsky 標準形を要求しないので，*Visit-CYK* よりも一般的な形で記述する必要があるが，スタック U, フラグ $Visited[\cdot]$ を用いる点や再帰手続きになる点など基本手続きは *Visit-CYK* と変わらない．また，支持グラフ抽出ルーチンの動作は

6 スタック用手続きとして，スタック U を空にする $ClearStack(U)$, スタック U にオブジェクト x を push する $PushStack(x, U)$, スタック U を pop して，pop されたオブジェクトを返す $PopStack(U)$ の 3 つを用意する．

パーザ備え付けの構文木出力ルーチンや構文木数え上げルーチンによく似ている．従って，支持グラフ抽出ルーチンを実装するときにはこれらのルーチンを基にすればよい．

3.4 グラフィカル EM アルゴリズム

提案手法による PCFG 訓練のメインルーチン *Learn-PCFG* は図 11 のようになる．2 つのサブルーチン *CYK-Parser* と *Extract-CYK* は先に説明した．本節では gEM アルゴリズムを実現する手続き *Graphical-EM* を記述する．

I-O アルゴリズムと同様，gEM アルゴリズムでも内側・外側確率という 2 つの確率値の計算が中心になる．各 $\tau \in O_\ell$ の内側確率，外側確率の値は $\mathcal{P}[\ell,\tau]$, $\mathcal{Q}[\ell,\tau]$ という配列変数に格納される．これは各部分支持グラフ $\tilde{\Delta}_\ell(\tau) = \langle \tau, \tilde{\psi}_\ell(\tau) \rangle$ によって保持される．また，$\tilde{\Delta}_\ell(\tau)$ は各局所パス $E \in \tilde{\psi}_\ell(\tau)$ ごとに配列変数 $\mathcal{R}[\ell,\tau,E]$ をもつ．また，配列変数 $\eta[A \to \zeta]$ に規則 $A \to \zeta$ の期待適用回数が格納される．*Graphical-EM* は内側確率を計算する *Get-Inside-Probs*, 外側確率と規則の期待適用回数を同時に計算する *Get-Expectations* という 2 つのサブルーチンをもつ．

Graphical-EM を図 12 に示す．*Graphical-EM* では，はじめにすべてのパラメタを初期化する（行 2）．そして，*Get-Inside-Probs*, *Get-Expectations*, パラメタの更新（行 7–8）をこの順に繰り返す．対数尤度 λ が収束したら（行 12），その時点でのパラメタ値 θ を推定値 θ^* と

```
1:  procedure Extract-CYK() begin
2:      for ℓ := 1 to N do begin
3:          Initialize all ψ̃_ℓ(·) to ∅ and all Visited[·] to NO;
4:          ClearStack(U);     /* スタックを初期化 */
5:          Visit-CYK(ℓ, S, 0, n_ℓ);     /* 各パーザ専用ルーチン; 三角行列右上隅から巡回 */
6:          for k := 1 to |U| do τ_k := PopStack(U);     /* スタック中の整列結果を順に取り出す */
7:          O_ℓ := ⟨τ_1, τ_2, ..., τ_|U|⟩
8:      end
9:  end.
```

```
1:  procedure Visit-CYK(ℓ, A, d, d') begin
2:      Put τ = A(d, d') and then Visited[τ] := YES;     /* 訪問を記録 */
3:      if d' = d + 1 and A(d, d')@w_d'(d, d') ∈ T^(ℓ)_{d,d'} then Add a set {A → w_d'} to ψ̃_ℓ(τ)
4:      else
5:          foreach A(d, d')@B(d, d'')C(d'', d') ∈ T^(ℓ)_{d,d'} do begin
6:              Add a set {A → BC, B(d, d''), C(d'', d')} to ψ̃_ℓ(τ);
7:              if Visited[B(d, d'')] = NO then Visit-CYK(ℓ, B, d, d'');     /* 再帰 */
8:              if Visited[C(d'', d')] = NO then Visit-CYK(ℓ, C, d'', d')     /* 再帰 */
9:          end;
10:     PushStack(τ, U)
11: end.
```

図 10　（上）支持グラフ抽出ルーチン *Extract-CYK*,（下）CYK パーザ用サブルーチン *Visit-CYK*.

```
1: procedure Learn-PCFG(C) begin
2:     CYK-Parser(C);    /* C の解析結果の WFST を生成 */
3:     Extract-CYK();    /* WFST から支持グラフを抽出 */
4:     Graphical-EM()    /* 支持グラフを参照しながらパラメタ θ を訓練 */
5: end.
```

図 11　PCFG 訓練のメインルーチン *Learn-PCFG*.

して終了する．$\mathcal{P}[\ell, S(0,n_\ell)]$ に文 $\mathbf{w}_\ell$ の生起確率 $P(\mathbf{w}_\ell)$ が格納されており，対数尤度の計算にはこの値を使う（行 4, 11）．図 13 にサブルーチン *Get-Inside-Probs*, *Get-Expectations* を示す．また，図 14 は支持グラフ上における各々の計算イメージである．

Get-Inside-Probs における内側確率 $\mathcal{P}[\ell,\tau]$ の計算は O_ℓ の最後尾の部分支持グラフから順に行なう．τ_k の部分支持グラフ $\tilde{\Delta}_\ell(\tau_k) = \langle \tau_k, \tilde{\psi}_\ell(\tau_k)\rangle$ $(k = 1 \ldots |O_\ell|)$ の各局所パス $E \in \tilde{\psi}_\ell(\tau_k)$ ではパス中の各ノードの確率積を計算し，$\mathcal{R}[\ell,\tau_k,E]$ に格納する（行 7–8, 図 14 (1)）．その際，基本ノード $A{\rightarrow}\zeta$ に対してパラメタ $\theta(A{\rightarrow}\zeta)$ を乗じ，中間ノード τ' に対して内側確率 $\mathcal{P}[\ell,\tau']$ を乗じる[7]（図 14 (2)）．最後に $\mathcal{R}[\ell,\tau_k,E]$ の和によって $\mathcal{P}[\tau_k]$ を計算する（行 10, 図 14 (3)）．

一方，*Get-Expectations* では *Get-Inside-Probs* とは逆に O_ℓ の先頭の部分支持グラフから順に計算を進める．はじめに配列変数 $\mathcal{Q}$ と η を初期化する．特に外側確率 $\mathcal{Q}[\ell,\cdot]$ について O_ℓ の先頭要素 $\tau_1 = S(0,n_\ell)$ のみを 1, 他は 0 にする点に注意する（行 5–6）．次に，ある $k = 1 \ldots |O_\ell|$ について τ_k の部分支持グラフ $\tilde{\Delta}_\ell(\tau_k)$ の局所パス E を考える（行 8）．更に，行 11 で外側確率 $\mathcal{Q}$ が書き換えられる $\tau' \in E$ を考える．また，行 11 の式で $\mathcal{Q}[\ell,\tau']$ に加算されるのは，E における τ' の局所的な外側確率（パス E に現れる τ' 以外のノードの確率積）と

7 節 3.2 で述べた支持グラフの 3 つ目の特徴より $\tau' = \tau_{k'}$ とおくと必ず $k < k'$ であることと，$\mathcal{P}$ の計算は O_ℓ の最後尾から順に行なわれることから，$\mathcal{P}[\ell,\tau']$ は参照されるとき既に計算済みになっている点に注意する．

```
1: procedure Graphical-EM() begin
2:     Initialize all parameters θ(A→ζ) such that P(w_ℓ|θ) > 0 for all ℓ = 1...N;
3:     Get-Inside-Probs();
4:     λ^(0) := Σ_{ℓ=1}^{N} log P[ℓ, S(0,n_ℓ)];
5:     repeat
6:         Get-Expectations();
7:         foreach (A→ζ) ∈ R do
8:             θ(A→ζ) := η[A→ζ] / Σ_{ζ':(A→ζ')∈R} η[A→ζ'];
9:         m += 1;
10:        Get-Inside-Probs();
11:        λ^(m) := Σ_{ℓ=1}^{N} log P[ℓ, S(0,n_ℓ)]
12:    until λ^(m) − λ^(m−1) becomes sufficiently small
13: end.
```

図 12　gEM アルゴリズムのメインルーチン *Graphical-EM*.

```
 1: procedure Get-Inside-Probs() begin
 2:   for ℓ := 1 to N do begin
 3:     Put O_ℓ = ⟨τ_1, τ_2, ..., τ_|O_ℓ|⟩;
 4:     for k := |O_ℓ| downto 1 do begin
 5:       foreach E ∈ ψ̃_ℓ(τ_k) do begin
 6:         R[ℓ, τ_k, E] := 1;
 7:         foreach τ' ∈ E do
 8:           if τ' = (A→ζ) then R[ℓ, τ_k, E] *= θ(A→ζ) else R[ℓ, τ_k, E] *= P[ℓ, τ']
 9:       end;   /* foreach E */
10:       P[ℓ, τ_k] := Σ_{E∈ψ̃_ℓ(τ_k)} R[ℓ, τ_k, E]
11:     end   /* for k */
12:   end   /* for ℓ */
13: end.
```

```
 1: procedure Get-Expectations() begin
 2:   foreach (A→ζ) ∈ R do η[A→ζ] := 0;
 3:   for ℓ := 1 to N do begin
 4:     Put O_ℓ = ⟨τ_1, τ_2, ..., τ_|O_ℓ|⟩;
 5:     for k := 2 to |O_ℓ| do Q[ℓ, τ_k] := 0;
 6:     Q[ℓ, τ_1] := 1;    /* τ_1 は特別に 1 に初期化 */
 7:     for k := 1 to |O_ℓ| do
 8:       foreach E ∈ ψ̃_ℓ(τ_k) do
 9:         foreach τ' ∈ E do
10:           if τ' = (A→ζ) then η[A→ζ] += Q[ℓ, τ_k] · R[ℓ, τ_k, E]/P[ℓ, S(0, n_ℓ)]
11:           else if P[ℓ, τ'] > 0 then Q[ℓ, τ'] += Q[ℓ, τ_k] · R[ℓ, τ_k, E]/P[ℓ, τ']
12:   end   /* for ℓ */
13: end.
```

図 13　*Graphical-EM* のサブルーチン：（上）内側確率を計算する *Get-Inside-Probs*，（下）外側確率および規則の期待適用回数を計算する *Get-Expectations*．

τ' の親部分木 τ_k の外側確率 $\mathcal{Q}[\ell, \tau_k]$ の積である[8]（図 14 (4)）．また，行 10 において，基本ノード $A \rightarrow \zeta$ に対しては局所パスの確率 $\mathcal{R}[\ell, \tau_k, E]$ と親部分木 τ_k の外側確率 $\mathcal{Q}[\ell, \tau_k]$ の積を文 $\mathbf{w}_\ell$ の生起確率 $P(\mathbf{w}_\ell)$ で割って[9] $\eta[A \rightarrow \zeta]$ に足し込む（図 14 (5)）．こうして $\eta[A \rightarrow \zeta]$ の内

8 $\tau' = \tau_{k'}$ とおくと，支持グラフの 3 つ目の特徴より必ず $k < k'$ が成り立つので，τ' は O_ℓ では常に τ_k より後ろに現れる．逆にいえば $k'' = k \ldots |O_\ell|$ なる部分支持グラフ $\tilde{\Delta}_\ell(\tau_{k''})$ では $\mathcal{Q}[\ell, \tau_k]$ の値は書き換えられることはなく，従って行 11 の式の右辺に現れる $\mathcal{Q}[\ell, \tau_k]$ は既に計算済みである．

9 *Graphical-EM* 行 2 のように，すべての $\ell = 1 \ldots N$ について $P(\mathbf{w}_\ell|\theta) > 0$ となる θ に初期化しているので，以降 θ の更新が行なわれても $P(\mathbf{w}_\ell|\theta) = 0$ となることはない．この事実は，gEM アルゴリズムにおける $\eta[r]$ の更新値と Fujisaki らの方法（式 14）における $\eta[r]$ の更新値が等しいと仮定したとき（これは付録 A で直観的に証明される），以下のように帰納的に証明される：まず m 回目の更新後のパラメタ $\theta^{(m)}$ の下で $P(\mathbf{w}_\ell|\theta^{(m)}) > 0$ が成り立つとする．すると，ある $\mathbf{r} \in \tilde{\psi}(\mathbf{w}_\ell)$ に対して $P(\mathbf{r}|\theta^{(m)}) > 0$ が成り立つ．そしてこの $\mathbf{r}$ に出現する任意の規則 $r \in \mathbf{r}$ について，$\sigma(r, \mathbf{r}) > 0$ が成り立つことと式 14 より，$\theta^{(m)}$ の下で $\eta[r] > 0$ となる．すると *Graphical-EM* 行 8 により更新後のパラメータ $\theta^{(m+1)}(r) > 0$ が保たれる．従って同じ $\mathbf{r}$ に対して $P(\mathbf{r}|\theta^{(m+1)}) > 0$ が成り立ち，これより $P(\mathbf{w}_\ell|\theta^{(m+1)}) > 0$ もまた成り立つ．以上で $P(\mathbf{w}_\ell|\theta^{(m)}) > 0 \Rightarrow P(\mathbf{w}_\ell|\theta^{(m+1)}) > 0$ が言えたので，パラメタを $P(\mathbf{w}_\ell|\theta^{(0)}) > 0$ なる $\theta^{(0)}$ に初期化すれば，以降の更新 $m = 1, 2, \ldots$ では必ず $P(\mathbf{w}_\ell|\theta^{(m)}) > 0$ である．（証明終）また，$P(\mathbf{w}_\ell|\theta) > 0$ とするために，現実的にはすべての規則 $r \in R$ について $\theta(r) > 0$ となる θ を選べば十分である．

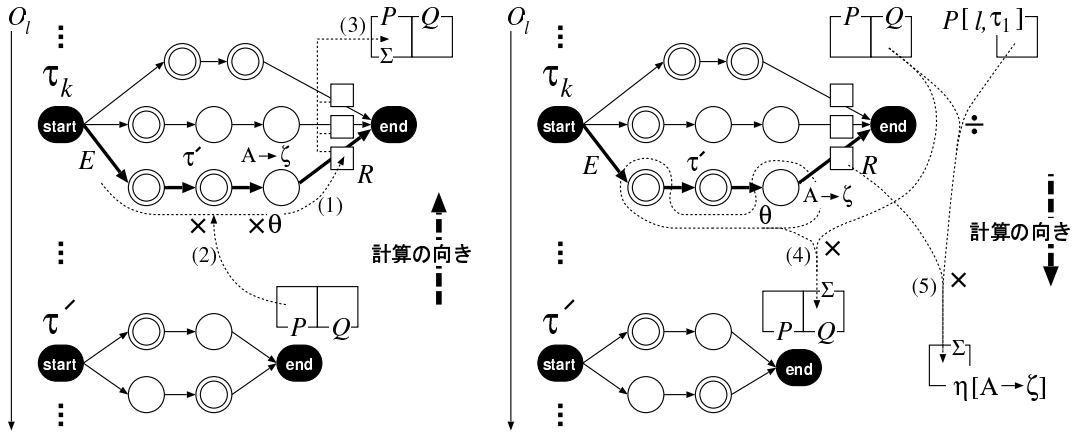

図 14　サブルーチン *Get-Inside-Probs*（左）と *Get-Expectations*（右）の計算イメージ.

容を書き換えていくと，*Get-Expectations* の終了時には $\eta[A \to \zeta]$ に $A \to \zeta$ の期待適用回数が格納されている．gEM アルゴリズムの計算は支持グラフの 1 つ目の特徴である支持グラフ Δ_ℓ の再帰的巡回（節 3.2）に基づいて正当化される．それを付録 A で示す.

一般に，EM アルゴリズムは尤度関数の山登りを行なうため局所的な最尤推定しか保証しない．従って訓練されたパラメタの質は初期パラメタ値に依存する．Lari と Young は HMM を利用して初期パラメタ値を与える方法を提案している (Lari and Young 1990)．最も簡便な解決法としては，初期パラメタをランダムに設定することと EM アルゴリズムを動作させることを h 回繰り返し，その中で収束時の対数尤度が最も高かった回の収束パラメタ値を訓練パラメタ値とする．この方法を以降では簡単に再出発法と呼ぶ.

3.5　予測構文木の計算

いったんパラメタ θ^* が訓練されたら，括弧なしであるテストコーパスの各文 $\mathbf{w}_\ell$ に対して $\mathbf{r}_\ell^* \stackrel{\text{def}}{=} \max_{\text{all } \mathbf{r}} P(\mathbf{r}|\mathbf{w}_\ell) = \max_{\text{all } \mathbf{r}} P(\mathbf{r}, \mathbf{w}_\ell) = \max_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} P(\mathbf{r})$ なる $\mathbf{r}_\ell^*$ を計算することができる．$\mathbf{r}_\ell^*$ に対応する構文木 t_ℓ^* を予測構文木（以下，単に予測木）という．この予測木 t_ℓ^* によって入力文 $\mathbf{w}_\ell$ に対する構文的曖昧性が解消される．ただし，$|\psi(\mathbf{w}_\ell)|$ は指数オーダなので，ここでも支持グラフに基づいて t_ℓ^* を計算する.

予測木 t_ℓ^* を計算する手続き *Predict* およびそのサブルーチン *Construct-Tree* を図 15 に示す．*Predict* はテストコーパス $\mathcal{C} = \langle \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N \rangle$ を受けとり，各 $\mathbf{w}_\ell$ に対する予測木中の部分木ラベルの集合 $\mathcal{L}(t_\ell^*)$ を $\mathcal{L}_\ell^*$ に格納する．*Predict* では，はじめにパーザ，支持グラフ抽出ルーチン，内側確率計算ルーチン *Get-Inside-Probs* の 3 つを走らせる（行 3）．次に，*Get-Inside-Probs* が計算した確率値 $\mathcal{R}[\ell, \tau, E]$ を参照しながら，$\delta[\ell, \tau]$ に最も確率の高い τ の

```
1: procedure Predict(C) begin
2:    Put N = |C|;
3:    CYK-Parser(C);   Extract-CYK();   Get-Inside-Probs();
4:    for ℓ := 1 to N do begin
5:       Put O_ℓ = ⟨τ_1, τ_2, ..., τ_{|O_ℓ|}⟩;
6:       for k := 1 to |O_ℓ| do δ[ℓ, τ_k] := argmax_{E∈ψ̃_ℓ(τ_k)} R[ℓ, τ_k, E];
7:       L*_ℓ := ∅;
8:       Construct-Tree(ℓ, τ_1)    /* 必ず τ_1 = S(0, n_ℓ) である. */
9:    end
10: end.
```

```
1: procedure Construct-Tree(ℓ, τ) begin
2:    foreach τ' ∈ δ[ℓ, τ] such that τ' = A(d, d') do begin
3:       Add τ' to L*_ℓ;
4:       Construct-Tree(ℓ, τ')
5:    end
6: end.
```

図 15　予測木計算ルーチン *Predict* とそのサブルーチン *Construct-Tree*.

局所パスを記録する（行 6）. 再帰手続き *Construct-Tree* では，支持グラフ Δ_ℓ の再帰的巡回に基づき，$\delta[\ell,\tau]$ 中のラベル $A(d,d')$ を $\mathcal{L}^*_\ell$ に追加する（行 3）ことで予測木を構築する．$\delta[\ell,\tau]$ に複数の局所パス候補を格納するように拡張すれば，生起確率上位 n 個の予測木が獲得できる.

3.6　計算量

節 2.4 の I-O アルゴリズムの計算量評価で述べたように，収束までのパラメタ更新回数は初期値に依存するため，1 回のパラメタ更新に要する計算量を gEM アルゴリズムの計算量とする．手続き *Graphical-EM* では **repeat** ループ内の計算量をはかればよい．まず，各 $\ell = 1 \ldots N$ について $O_\ell = \langle \tau_1^{(\ell)}, \tau_2^{(\ell)}, \ldots, \tau_{|O_\ell|}^{(\ell)} \rangle$ とおく．*Graphical-EM* に呼び出される *Get-Inside-Probs* はその内部処理において $k = 1 \ldots |O_\ell|$ について $|\tilde{\psi}_\ell(\tau_k^{(\ell)})|$ の要素を一回ずつ訪れることから，

$$\mu_{\mathrm{num}} \stackrel{\mathrm{def}}{=} \max_{\ell=1\ldots N} \sum_{k=1}^{|O_\ell|} \left|\tilde{\psi}_\ell(\tau_k^{(\ell)})\right| \tag{16}$$

$$\mu_{\mathrm{maxsize}} \stackrel{\mathrm{def}}{=} \max_{E:\ \ell=1\ldots N,\ k=1\ldots|O_\ell|,\ E\in\tilde{\psi}_\ell(\tau_k^{(\ell)})} |E| \tag{17}$$

を導入すると，手続き *Get-Inside-Probs* の計算量は $O(\mu_{\mathrm{num}}\mu_{\mathrm{maxsize}}N)$ である．同様に手続き *Get-Expectations* においても $O(\mu_{\mathrm{num}}\mu_{\mathrm{maxsize}}N)$ の計算量を要する.

I-O アルゴリズムの計算量評価と同様，訓練コーパス $\mathcal{C}$ に対して最長の文の長さを L とし，非終端記号集合 V_{n}, 終端記号集合 V_{t} を固定する．我々は Chomsky 標準形を満たす文法に対する最悪計算量を考える．そのために，まず Chomsky 標準形を満たす最大の PCFG として節 2.4 式 6 で導入した規則集合 $R_{\max}$ を考える．このとき，$A \in V_{\mathrm{n}}$, $0 \le d, d' \le L$, $d+2 \le d'$

なる A, d, d' について下が成り立つ（$d' = d+1$ の場合は無視できる）：

$$\tilde{\psi}_\ell(A(d,d')) = \big\{\{A \to BC,\ B(d,d''),\ C(d'',d')\} \bigm| B,C \in V_{\rm n},\ d < d'' < d'\big\}. \qquad (18)$$

$|O_\ell| = \big|\{A(d,d') \mid A \in V_{\rm n},\ 0 \le d < d' \le L\}\big| = O(|V_{\rm n}|L^2)$ かつ $\big|\tilde{\psi}_\ell(\tau)\big| = O(|V_{\rm n}|^2 L)$ が成り立ち，定義より $\mu_{\rm num} = O(|V_{\rm n}|^3 L^3)$ となる．同様に定義より $\mu_{\rm maxsize} = 3 = O(1)$ である．また，$\theta(A \to \beta)$ の更新に要する計算量は $O(|R_{\max}|)$ だが，$|R_{\max}| = O(|V_{\rm n}|^3)$ なので無視できる．以上より手続き *Graphical-EM* の **repeat** ループ内の計算量は $O(|V_{\rm n}|^3 L^3 N)$ である．以上より gEM アルゴリズムの最悪計算量は I-O アルゴリズムと同じ $O(|V_{\rm n}|^3 L^3 N)$ である．

Chomsky 標準形を仮定したとき，CYK パーザ *CYK-Parser* と支持グラフ抽出ルーチン *Extract-CYK* の最悪計算量は EM の一更新ステップの最悪計算量と同じ $O(|V_{\rm n}|^3 L^3 N)$ である．ただ，EM アルゴリズムでは更新ステップを数 10 から数 100 回繰り返すのが通常なので *Extract-CYK* が訓練全体に占める割合は小さい．同様に Chomsky 標準形を仮定したとき，一つの文に対する生起確率計算，予測木の計算いずれの計算量も $O(|V_{\rm n}|^3 L^3)$ である ($N = 1$)．また，式 5 の形をした部分木の親子対を構成要素とする WFST をもつパーザ（例えば CYK や GLR）では，抽出される O_ℓ, $\tilde{\psi}_\ell$ は全く同じになるので，提案手法の計算量は組み合わせたパーザによる差はない．Earley パーザを用いた場合に関する評価は付録 B に示す．

4　訓練時間に関する実験

我々は現実的な文法に対しては I-O アルゴリズムに比べて EM 学習が大幅に高速化される（提案手法の**特長 2**）ことを示すため，ATR 対話コーパス (SLDB) でパラメタ推定に要する計算時間（訓練時間と呼ぶ）を計測した．対象 PCFG の元になる CFG は 860 規則から成る，田中らが開発した音声認識用日本語文法 (田中, 竹澤, 衛藤 1997) に手が加えられたものである．以降ではこの CFG を G^* で参照する．ATR 対話コーパスもこの文法に対応して手が加えられている．G^* は品詞を細分化したカテゴリを終端記号とした CFG であり，非終端記号数 173, 終端記号数 441 である．ATR 対話コーパス中の文では，(実際の単語ではなく）上記カテゴリの列を対象とした．文長は平均 9.97, 最短 2, 最長 49 である．また，G^* の規則集合 R^* は Chomsky 標準形ではないので，GLR パーザとの組合せを採用した[10]．

本論文の実験では G^* が与えられた場合の訓練時間を提案手法と I-O アルゴリズムの間で比較する．ただし，I-O アルゴリズムにおいては節 2.4 で記述したものを用い，そこで参照される規則集合 R には，全ての終端・非終端記号の組合せから成る Chomsky 標準形の規則集合

10 具体的には，東京工業大学 田中・徳永研究室で開発・公開されている MSLR (Morphological and Syntactic LR) パーザに支持グラフ抽出ルーチンと gEM アルゴリズムのルーチンを連結した．MSLR パーザは `http://tanaka-www.cs.titech.ac.jp/pub/mslr/` で公開されている．MSLR パーザは形態素解析と構文解析を同時に行なう機能を有するが，今回の実験では構文解析機能のみを使用した．MSLR パーザを含め，実験で用いたプログラムはすべて C 言語で実装されている．C コンパイラは gcc 2.8.1 を使用した．また，実験で使用した計算機の CPU と OS はそれぞれ Sun UltraSPARC-II 296 MHz と Solaris 2.6 である．

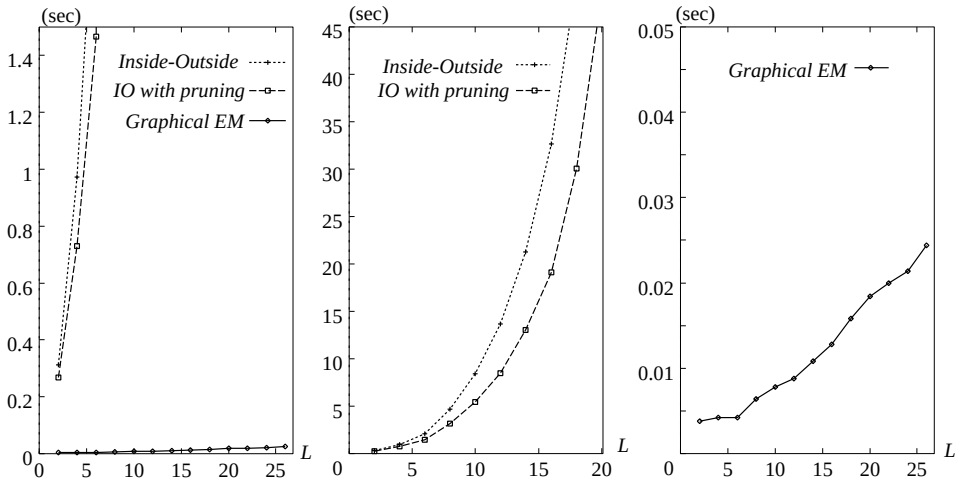

図 16　(左) Inside-Outside アルゴリズムとその枝刈り版，および gEM アルゴリズムにおける更新時間 (sec) の変化，(中央) 縦軸を縮小したもの，(右) 縦軸を拡大したもの．

$R_{\max}$ ではなく，G^* の規則集合 R^* を用いる点に注意する[11]．我々は文長 L を変化させたときにパラメタを一回更新するのに要する計算時間（更新時間と呼ぶ）が変化する様子を比較する．まず，我々は ATR コーパス $\mathcal{C}$ の中で文長 $L-1$ と L の文をグループ化し，各々から無作為に取り出した 100 文を $\mathcal{C}_L$ とする ($L = 2, 4, \ldots 26$)[12]．そして，各 $\mathcal{C}_L$ を一つの訓練コーパスとし，各々に対して更新時間を計測する．I-O アルゴリズムは Chomsky 標準形でしか動作しないので，あらかじめ G^* を Chomsky 標準形に変換した．その結果 860 規則が 2,308 規則（非終端記号数 210, 終端記号数 441）の文法になった．

更新時間を計測した結果を図 16 左に示す．縦軸が更新時間 (sec)，横軸 L が使用した訓練コーパス $\mathcal{C}_L$ を表す．“*Inside-Outside*” は I-O アルゴリズムの更新時間，“*IO with pruning*” は (北 1999) で説明されている，I-O アルゴリズムの外側確率の計算において無駄な計算部分を枝刈りするように改良したものである．これを以下では枝刈り版 I-O アルゴリズムと呼ぶ．“*Graphical EM*” は gEM アルゴリズムの更新時間を示す．また，変化の様子を見やすくするために，図 16 左の縦軸を拡大，縮小したものをそれぞれ図 16 中央，図 16 右に示す．図 16 中央において gEM アルゴリズムの更新時間は見にくいため省略した．

図 16 左のグラフから分かるように，gEM アルゴリズムは I-O アルゴリズムやその枝刈り版に比べてはるかに高速な計算が行なわれていることが分かる．また，図 16 中央のグラフから分かるように I-O アルゴリズムは理論値どおり L^3 の曲線を描く．枝刈り版 I-O アルゴリズムは枝刈りした分高速であるものの，仮説駆動型である（$L \times L$ の三角行列の全要素を走査す

11 当然，R^* を用いた場合の I-O アルゴリズムは，その文法制約のため $R_{\max}$ を用いた場合より高速になる．
12 長さ 27 以上の 176 文（全体の 1.6 %）はデータ不足のため，実験では考慮しなかった．

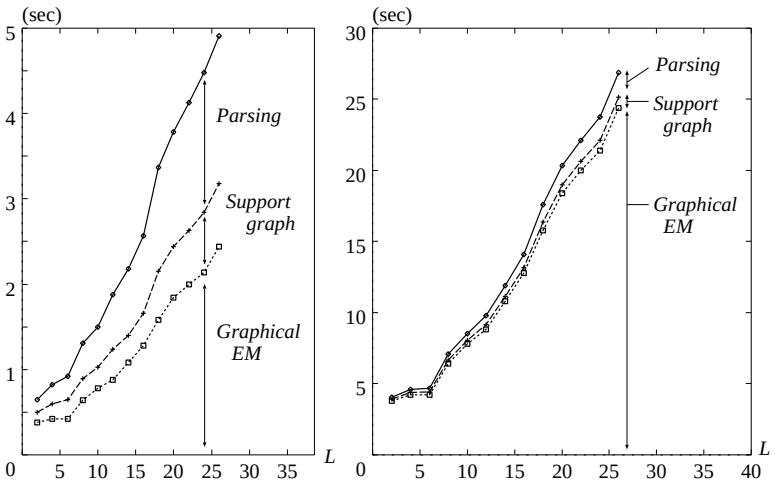

図 17　訓練時間全体に占める各過程の処理時間の内訳.（左）再出発なしの場合 ($h = 1$),（右）再出発回数 $h = 10$ の場合.

る）点は変わらないので，枝刈りが最も効率良く行なわれた場合でも L^2 を下回ることはない. 収束まで数 100 回の更新を要すること，および再出発法を採用することを考慮すると，$L = 20$ を越える訓練コーパス $\mathcal{C}_L$ に 対して I-O アルゴリズムおよびその枝刈り版を収束するまで動作させるのは現実的ではない．それに対し，提案手法では $L = 2, 4, \ldots, 26$ の範囲では L に対してほぼ線形に計算できており（図 16 右），最悪計算量 $O(|V_{\mathrm{n}}|^3 L^3)$ とは大きな差があることが分かった．これは文法の制約により，WFST に格納される部分木の数が抑えられたためと考えられる．ATR コーパスにおける文長平均 9.97 に近い $L = 10$ では I-O アルゴリズムに対しておよそ 1,000 倍（枝刈り版に対してはおよそ 700 倍）の速度向上が得られた.

良質なパラメタを得る目的で再出発法（節 3.4）を採用すると，訓練時間の内訳は

$$\begin{aligned}(\text{全体の訓練時間}) \quad &= \quad (\text{構文解析時間}) + (\text{支持グラフ抽出に要する時間}) \\ &\qquad + (\text{gEM アルゴリズム実行時間}), \\ (\text{gEM アルゴリズム実行時間}) \quad &= \quad (\text{更新時間}) \times (\text{収束までの更新回数}) \times (\text{再出発回数 } h).\end{aligned}$$

となる．先に述べた文長毎の訓練コーパス $\mathcal{C}_L$ ($L = 2, 4, \ldots, 26$) を使って，訓練時間の内訳（構文解析時間，支持グラフ抽出時間，gEM 実行時間）を計測した．その結果を図 17 に示す．横軸が L, 縦軸が処理時間 (sec) である．図 17（左）は再出発なし ($h = 1$) の場合，図 17（右）は再出発回数 $h = 10$ の場合である．また，収束までの更新回数はコーパス $\mathcal{C}_L$ によって異なるため，ここでは 100 に固定した．構文解析時間 (“*Parsing*”)，支持グラフ抽出時間 (“*Support graph*”)，gEM 実行時間 (“*Graphical EM*”) はいずれも文長 L に対してほぼ線形になっていることが分かる．更に図 17（右）より，再出発法を採用した場合は構文解析時間と支持グラフ抽

出時間が訓練時間全体に占める割合は非常に小さい．構文解析と支持グラフ抽出は再出発の度に繰り返す必要がないからである．構文解析と支持グラフ抽出を gEM アルゴリズムの前処理と捉えれば，わずかな前処理（図 17）で大きな速度向上（図 16）が得られているということができ，構文解析と EM 学習を分離したメリットが現れている．

5　PCFG の拡張文法の EM 学習

これまで PCFG に文脈依存性を採り入れたモデル（PCFG の拡張文法と呼ぶ）が数多く提案されているが，Charniak らの疑似確率文脈依存文法 (pseudo probabilistic context-sensitive grammars) (Charniak and Carroll 1994) を除けば EM アルゴリズムを具体的に記述した文献は見当たらない．本節では，提案手法が PCFG の拡張文法に対する多項式オーダの EM アルゴリズムを包含する（提案手法の**特長 3**）ことを示すため，一例として Kita らの規則バイグラムモデル (Kita et al. 1994) を取り上げ，その多項式オーダの EM アルゴリズムを導出する．

5.1　規則バイグラムモデルとその EM アルゴリズム

まず，我々は PCFG のときと同様に導出戦略は最左導出に固定する．規則バイグラムモデルでは，節 2.1 で述べた PCFG の「規則選択は他と独立」という仮定の代わりに，「規則選択は直前の選択のみに依存する」という仮定をおく．従って，規則バイグラムモデルでは PCFG では扱えなかった文脈依存性も若干考慮できる．この仮定の下で適用規則列 $\mathbf{r}$ の出現確率は

$$P(\mathbf{r}) = \theta(r_1 \mid \#) \prod_{k=2}^{K} \theta(r_k \mid r_{k-1}) \tag{19}$$

と計算される．$\#$ は境界を表すマーカ，$\theta(r \mid r')$ は各規則 $r \in R$ に付与されるパラメタである ($r' \in R \cup \{\#\}$)．各 $A \in V_{\mathrm{n}}$, $r \in R \cup \{\#\}$ に対し $\sum_{\zeta:(A \to \zeta) \in R} \theta(A \to \zeta \mid r) = 1$ が成り立つ．(Kita et al. 1994) で示された，括弧なしコーパス $\mathcal{C} = \langle \mathbf{w}_1, \ldots, \mathbf{w}_N \rangle$ に基づく $\theta(r_k|r_{k-1})$ の推定式は式 20 のとおりである．適用規則列 $\mathbf{r}$ に対して，$\sigma(r, r'; \mathbf{r})$ は $\mathbf{r}$ において r' が r の直後に出現する頻度を表す．定義より明らかに $\sum_{r' \in R} \sigma(r, r'; \mathbf{r}) = \sigma(r; \mathbf{r})$ が成り立つ．

$$\theta(r_k|r_{k-1}) := \left(\sum_{\ell=1}^{N} \frac{\sum_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} \sigma(r_{k-1}, r_k; \mathbf{r})}{|\psi(\mathbf{w}_\ell)|} \right) \Bigg/ \left(\sum_{\ell=1}^{N} \frac{\sum_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} \sigma(r_{k-1}; \mathbf{r})}{|\psi(\mathbf{w}_\ell)|} \right) \tag{20}$$

ところが式 9, 14 から類推できるように，EM アルゴリズムの考えに基づく更新式は次のようになる ($m = 1, 2, \ldots$)．つまり式 20 は相対頻度法，EM アルゴリズムのいずれにもなっていない．

$$\theta^{(m+1)}(r_k|r_{k-1}) := \left(\sum_{\ell=1}^{N} \frac{\sum_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} P(\mathbf{r}|\theta^{(m)}) \sigma(r_{k-1}, r_k; \mathbf{r})}{P(\mathbf{w}_\ell|\theta^{(m)})} \right) \Bigg/ \left(\sum_{\ell=1}^{N} \frac{\sum_{\mathbf{r} \in \psi(\mathbf{w}_\ell)} P(\mathbf{r}|\theta^{(m)}) \sigma(r_{k-1}; \mathbf{r})}{P(\mathbf{w}_\ell|\theta^{(m)})} \right) \tag{21}$$

式 21 の更新式により（局所）最尤推定は実現されるが，これまで述べてきたように一般に $|\psi(\mathbf{w})|$ は文長 $|\mathbf{w}|$ に対して指数オーダになるため，式 21 は現実時間で計算できない．

一方，提案手法に基づき，式 21 と等価な規則バイグラムモデルの多項式オーダの EM アルゴリズムを導出することができる．次節でアルゴリズムを記述するが，その前にいくつかの記号を導入する．まず，次のような文 $\mathbf{w}$ の最左導出列 $\mathbf{r}$ を考える:

$$S \overset{*}{\Rightarrow} \cdots \overset{r}{\Rightarrow} \mathbf{w}_{0,d} A\xi \overset{r''}{\Rightarrow} \mathbf{w}_{0,d}\zeta\xi \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \mathbf{w}_{0,d}\zeta'\xi \overset{r'}{\Rightarrow} \mathbf{w}_{0,d}\mathbf{w}_{d,d'}\xi (= \mathbf{w}_{0,d'}\xi) \overset{*}{\Rightarrow} \mathbf{w} \quad . \tag{22}$$

を考える．式 22 において r は A を展開する直前に適用された規則，r' は導出 $A \overset{*}{\Rightarrow} \mathbf{w}_{d,d'}$ で用いられた最後の規則である．r と r' を考慮した，$\mathbf{w}_{d,d'}$ を統治する部分木ラベルを $A(d,d'|r,r')$ で表す．また，式 22 において r' を $last(A,d,d';\mathbf{r})$ で参照し，r の次に適用された規則 r'' を $A \to \zeta|_r$ で参照する．前節で述べた $\theta(A \to \zeta|r)$ の確率で $A \to \zeta|_r$ が適用される．また，$\mathbf{w}$ の構文木中の部分木 $A(d,d')$ を導出するとき最後に使われた規則の集合を $last(A,d,d';\mathbf{w}) \overset{\text{def}}{=} \bigcup_{\mathbf{r}\in\psi(\mathbf{w})} last(A,d,d';\mathbf{r})$ と定める．

5.2　グラフィカル EM アルゴリズムの適用

ここでは CYK パーザと組み合わせた場合の規則バイグラムモデルの EM 学習法を示す．規則バイグラムモデルを対象にする場合，パーザに新たな変更を加える必要はない．また，gEM アルゴリズムもその汎用性により，対象とする確率値の意味が変わるだけで制御構造に変化はない．従って，我々は支持グラフ抽出ルーチンを変更するだけである．例えば，図 5 の $t2$ にでは次のような関係 $\tilde{\psi}_\ell$ が得られる．

$$\tilde{\psi}_\ell(\mathtt{VP}(1,5 \mid \mathtt{ADV}\to\text{急いで},\ \mathtt{V}\to\text{見た})) = \left\{ \left\{ \mathtt{VP}\to\mathtt{PP\ V}|_{\mathtt{ADV}\to\text{急いで}},\ \mathtt{PP}(1,4 \mid \mathtt{VP}\to\mathtt{PP\ V},\ \mathtt{P}\to\text{を}),\ \mathtt{V}(4,5 \mid \mathtt{P}\to\text{を},\ \mathtt{V}\to\text{見た}) \right\} \right\}$$

節 3.1 で示した PCFG の場合に比べて，部分木ラベル $A(d,d')$ が，その導出直前に適用された規則と自身の導出において最後に適用された規則の組（“|” 記号の後ろ）によって細分化されており，この細分化によって文脈依存性が表現される．

規則バイグラム用の支持グラフ抽出ルーチン *Extract-CYK-RB* とそのサブルーチン *Visit-CYK-RB* をそれぞれ図 18, 図 19 に示す．*Visit-CYK-RB*(ℓ,r,A,d,d') は $\mathbf{w}_\ell$ の構文木中の部分木 $A(d,d')$ を訪問し，大域的配列変数 $Last[A(d,d')]$ に $last(A,d,d';\mathbf{w}_\ell)$ を格納する再帰手続きである．後は gEM アルゴリズム（手続き *Graphical-EM*, *Get-Inside-Probs*, *Get-Expectations*）において $A\to\zeta, \theta(A\to\zeta), \eta[A\to\zeta]$ を各々 $A\to\zeta|_r, \theta(A\to\zeta|r), \eta[A\to\zeta|r]$ といった規則バイグラム用の確率値，期待値に書き換え，*Graphical-EM* 行 7–8 と *Get-Expectations* 行 2 の **foreach** ループに “**foreach** $r \in R$” ループを重ねるだけでよい．

次に，規則バイグラム用 EM アルゴリズムの最悪計算量を評価する．$R_{\max}$ を考えたとき，

```
1: procedure Extract-CYK-RB() begin
2:   for ℓ := 1 to N do begin
3:     Initialize all ψ̃_ℓ(·) to ∅ and all Visited[·,·] and Last[·] to NO;
4:     ClearStack(U);
5:     Visit-CYK-RB(ℓ, S, 0, n_ℓ, #);    /* # は境界を表すマーカ．  */
6:     for k := 1 to |U| do τ_k := PopStack(U);
7:     Prepare some τ_0;
8:     ψ̃_ℓ(τ_0) := {{S(0, n_ℓ|#, r)} | r ∈ Last[S(0, n_ℓ)]};
9:     O_ℓ := ⟨τ_0, τ_1, τ_2, ..., τ_|U|⟩
10:  end
11: end.
```

図 18　規則バイグラム用支持グラフ抽出ルーチン *Extract-CYK-RB*

最悪計算量は $O(|V_{\rm n}|^{12}L^3N)$ となる[13]．これは非常に大きなオーダであるが，文長 L に対して 3 乗のオーダである点は I-O アルゴリズムと変わらない．また，節 4 の実験結果は PCFG に対する現実の計算時間と最悪時の計算時間 $O(|V_{\rm n}|^3L^3)$ に大きな差があることを示しており，これは規則バイグラムモデルでも成り立つと考えられる．実際森らは，節 4 の実験で用いた CFG G^* に対し本節で述べた方法を適用した結果，規則バイグラムの EM 学習におけるパラメタ更新時間が PCFG（図 16 右）の 1.5 倍程度で収まることを報告している (森, 亀谷, 佐藤 2000)．

6　関連研究

まず，Magerman らの $\mathcal{P}$earl (Magerman and Marcus 1991) およびその後継である $\mathcal{P}$icky (Magerman and Weir 1992), また Stolcke の確率的 Earley パーザ (Stolcke 1995) をはじめ，確率的パーザが多く提案されている．しかし，それらの多くは文法構造 G とパラメタ θ が与えられていることを前提としており，Stolcke を除けば PCFG（もしくはその拡張文法）の EM 学習について具体的に記述しているものは少ない．

Chomsky 標準形でない PCFG の訓練法としては，Kupiec の方法 (Kupiec 1992) と先述の Stolcke の確率的 Earley パーザによる訓練が挙げられる．Kupiec の方法は PCFG を再帰遷移ネットワークと捉え，拡張した trellis 図に基づき訓練を行なうものである．しかし，仮説駆動型である点は I-O アルゴリズムと変わらない．また，提案手法で用いる WFST は，CFG に基

13 まず，$A \in V_{\rm n}$, $0 \le d < d' \le L$ かつ $d+2 \le d'$ なる A, d, d' および，$r, r' \in R$ について

$$\tilde{\psi}_\ell(A(d,d'|r,r')) = \left\{ \left\{ A \to BC|_r,\ B(d,d''|A \to BC, r''),\ C(d'',d'|r'',r') \right\} \;\middle|\; \begin{array}{l} B, C \in V_{\rm n},\ d < d'' < d', \\ r'' \in last(B,d,d'') \subseteq R \end{array} \right\}$$

となるような $A(d,d'|r,r')$ が O_ℓ 中に出現する ($\ell = 1 \ldots N$)．$\left|\tilde{\psi}_\ell(\tau)\right| = O(|V_{\rm n}|^2L|R|)$ であるのは明らかである．また，O_ℓ は $\left\{ A(d,d'|r,r') \;\middle|\; A \in V_{\rm n},\ 0 \le d < d' \le L,\ r, r' \in R \right\}$ の部分集合を並べたものであるから，$|O_\ell| = O(|V_{\rm n}|L^2|R|^2)$ となる．定義より $\mu_{\rm num} = O(|V_{\rm n}|^3L^3|R|^3)$, $\mu_{\rm maxsize} = O(1)$ であり，さらに最悪の場合 $R = R_{\max}$ を考えると gEM アルゴリズムの計算量は $O(|V_{\rm n}|^3L^3|R_{\max}|^3N) = O(|V_{\rm n}|^{12}L^3N)$ となる．

```
1:  procedure Visit-CYK-RB(ℓ, A, d, d′, r) begin
2:      Visited[A(d, d′), r] := YES;
3:      if d′ = d + 1 and A(d, d + 1)@w^(ℓ)_{d,d+1} ∈ T^(ℓ)_{d,d+1} then begin
4:          Add a set {A→w^(ℓ)_{d,d+1}|_r} to ψ̃_ℓ(A(d, d + 1|r, A→w^(ℓ)_{d,d+1}));
5:          Last[A(d, d′)] := {A→w^(ℓ)_{d,d+1}}
6:      end
7:      else begin
8:          Last[A(d, d′)] := ∅;
9:          foreach A(d, d′)@B(d, d″)C(d″, d′) ∈ T^(ℓ)_{d,d′} do begin
10:             if Visited[B(d, d″), A→BC] = NO then Visit-CYK-RB(ℓ, B, d, d″, A→BC);
11:             foreach r″ ∈ Last[B(d, d″)] do begin
12:                 if Visited[C(d″, d′), r″] = NO then Visit-CYK-RB(ℓ, C, d″, d′, r″);
13:                 foreach r′ ∈ Last[C(d″, d′)] do begin
14:                     Add a set {A→BC|_r, B(d, d″|A→BC, r″), C(d″, d′|r″, r′)}
15:                         to ψ̃_ℓ(A(d, d′|r, r′));
16:                     PushStack(A(d, d′|r, r′), U)
17:                 end
18:             end;   /* foreach r″ */
19:             Last[A(d, d′)] := Last[A(d, d′)] ∪ Last[C(d″, d′)]
20:         end   /* foreach A@BC */
21:     end   /* else */
22: end.
```

図 19　規則バイグラム用支持グラフ抽出ルーチンのサブルーチン *Visit-CYK-RB*.
記述短縮のため，ここでは異なる引数 r（直前の適用規則）の呼び出しについて $Last[A(d, d')]$ の計算を重複して行なうものを示す.

づく構文解析にとって本質的なデータ構造であることから，本手法は trellis 図に基づく Kupiec の方法よりも簡潔で理解しやすいものと考える．一方，ε 規則やサイクル $A \stackrel{+}{\Rightarrow} A$ が存在しない PCFG に対して，Stolcke の方法は我々の枠組で Earley パーザと gEM アルゴリズムを組み合わせた場合と等価である．すなわち，このような PCFG に対して我々の枠組は Stolcke の方法の一般化になっている．Stolcke の方法との対応づけを付録 B に示す．また，Stolcke は PCFG の拡張文法については言及していない．ε 規則やサイクル $A \stackrel{+}{\Rightarrow} A$ をもつ PCFG に対する訓練法を考えるのは今後の課題であるが，提案手法は現段階においても充分実用的である.

Peraira と Schabes は部分もしくは完全括弧コーパスから PCFG の文法構造を学習する方法を提案し，学習された文法構造とパラメタの質が括弧なしコーパスからの学習に比べ大きく向上することを実験的に示した (Pereira and Schabes 1992)．我々の枠組でも，括弧づけされた文に対し，括弧の制約を満たす構文木のみを出力する機能をもつパーザを用意すれば[14]，支持グラフ抽出ルーチン，gEM アルゴリズムに何の変更も加えることなく括弧つきコーパスからの

14 節 4 で用いた MSLR パーザはそのような機能を備えている.

訓練が可能になる．変更の必要がないのは，我々が最終的な構文木情報（すなわちWFST）のみを参照するためである．また，完全に括弧づけされた訓練コーパスに対しgEMアルゴリズムの計算量はPereiraとSchabesの方法と同じオーダ$O(|V_{\rm n}|^3LN)$であることも容易に分かる[15]．

本論文の手法は文法構造(CFG)が与えられていることを前提としているが，人間が精密な文法を記述するのに多くの手間を費やすことを考えると，文法構造の自動学習は重要な課題である．先述したように，Lariと Youngは非終端記号集合$V_{\rm n}$と終端記号集合$V_{\rm t}$をあらかじめ定めた上で先述した$R_{\max}(V_{\rm n},V_{\rm t})$を考え，I-Oアルゴリズムを走らせ，推定後にパラメタ値が小さい規則を除去する方法を提案した(Lari and Young 1990)．また，先述したPereira & Schabesの学習法(Pereira and Schabes 1992)も括弧づけコーパスからの文法学習と捉えることができる．しかし，一般にEMアルゴリズムは局所的な最尤推定値しか保証しないため，学習される文法の質はパラメタの初期値に大きく依存し，文法学習を困難にしている．それに対し，HMMでは逐次状態分割(SSS)法(鷹見 嵯峨山 1993)やモデル選択規準に基づくHMMの構造探索法(池田 1995)のように，パラメタ訓練と構造探索を分離し，これらを交互に繰り返して良質なモデル構造を得る方法が提案されている．どちらの手法もパラメタ訓練ステップではモデル（文法）構造が与えられるので，上記手法をPCFGの構造学習に一般化したとき[16]，本論文で示した高速化が有効に働くものと期待する．

本論文で示したgEMアルゴリズムは最小モデル意味論の確率的一般化である分布意味論(Sato 1995)に基づく確率的な論理プログラミング言語PRISM(Sato and Kameya 1997)における高速EM学習のために提案されたものである(Kameya and Sato 2000)．そこではOLDT探索(Tamaki and Sato 1986)とgEMアルゴリズムを連結するが，本論文の手法はPCFGおよびその拡張文法用にOLDTをパーザに置き換えて特殊化を図ったものである．OLDT探索を構文解析に用いることも可能だが，OLDT探索はトップダウン（仮説駆動）探索であるので，LR表へのコンパイル・ボトムアップ探索を利用するGLRパーザの方が現実文法ではより高速である．得られる支持グラフはまったく同じなのでgEMアルゴリズムの計算時間は変わらない．

7　まとめ

文法構造が与えられていることを前提に，確率文脈自由文法(PCFG)を括弧なしコーパスから訓練するための一般的な枠組を提案し，従来法であるInside-Outsideアルゴリズムの一般化

15 Pereira と Schabes は$O(L)$としか明記していないが，彼らが提示したアルゴリズムより$R_{\max}(V_{\rm n},V_{\rm t})$に対して$O(|V_{\rm n}|^3LN)$となることは明らかである．また，gEMアルゴリズムで$O(|V_{\rm n}|^3LN)$であることは次のように示される：まず，$\mathbf{w}_\ell$に与えられた括弧集合$\mathcal{B}(\mathbf{w}_\ell)$のサイズは（Pereira と Schabes も述べているように）$O(|\mathbf{w}_\ell|)$である．与えられた$\mathcal{B}(\mathbf{w}_\ell)$と一致しない部分木は最終的な構文木にはならない（すなわちO_ℓの要素にはならない）ので，$\forall\ell=1\ldots N$について$|O_\ell|=|\{A(d,d')\mid A\in V_{\rm n}\text{ かつ }(d,d')\in\mathcal{B}(\mathbf{w}_\ell)\}|=O(|V_{\rm n}|\cdot|\mathbf{w}_\ell|)=O(|V_{\rm n}|L)$である．また，式18において，$\mathcal{B}(\mathbf{w}_\ell)$に一致する$d'$は高々1つであるから，すべての$\ell=1\ldots N$について$|\tilde{\psi}_\ell(A(d,d'))|=O(|V_{\rm n}|^2)$である．従って，式16より$\mu_{\rm num}=O(|V_{\rm n}|^3L)$である．前の議論と同様に$\mu_{\rm maxsize}=O(1)$であるから，gEMアルゴリズムにおいて一回のパラメタ更新に要する計算量は$O(|V_{\rm n}|^3LN)$である．

16 もちろん，そのときはパラメタ訓練ステップが更に構文解析ステップとgEMアルゴリズムステップに分離される．

と（現実文法における）高速化を同時に実現した．提案手法では PCFG の訓練過程を構文解析と EM 学習を分離し，パーザが記録する WFST から訓練文と関係のある部分木構造のみを抽出してから EM 学習することにより，仮説駆動型であった Inside-Outside アルゴリズムの計算効率上の欠点を克服した．また，従来知られてきた構文解析の高速化技術が PCFG の訓練にそのまま反映される．更に，提案手法を実装し，ATR 対話コーパスにおける訓練時間を計測したところ，Inside-Outside アルゴリズムに比べコーパス平均文長においておよそ 1,000 倍の速度向上が得られることを確認した．また，提案手法の一般性に基づき，文脈依存性を考慮した PCFG の拡張文法（北らの規則バイグラムモデル）の多項式オーダの EM アルゴリズムを導出した．加えて，確率 Earley パーザによる Stolcke の EM 学習法や Pereira と Schabes らによる部分括弧つきコーパスからの学習法も提案手法の枠組で扱えることを示し，提案手法が CFG に基づく確率言語モデルの訓練手法を広くカバーしていることを明らかにした．今後の課題としては，PCFG の拡張文法を用いた実験や文法構造の学習，また支持グラフと gEM アルゴリズムの一般性を利用して，Inui らによって再定式化された確率 GLR モデル (Inui, Sornlertlemvanich, Tanaka, and Tokunaga 1998) の効率的な EM アルゴリズムの導出を試みるのも興味深い．

謝辞

実験に用いた ATR コーパス，日本語文法の改訂版は，東京工業大学 田中・徳永研究室のご厚意により提供頂きました．記して感謝致します．また，同研究室白井清昭助手には上記コーパス・文法に関する情報やテキスト処理プログラムの提供，文献紹介など貴重なご助力を頂きました．重ねて感謝申し上げます．また，東京工業大学 佐藤泰介研究室の上田展久氏には文献紹介を含め，数多くの有益なコメントを頂きました．感謝致します．なお，本研究の一部は平成 11 年度 科学研究費補助金 特定領域研究 (A) 「発見科学」の補助を受けています．

参考文献

Baker, J. K. (1979). “Trainable grammars for speech recognition.” In *Proc. of the Spring Conf. of the Acoustical Society of America*, pp. 547–550.

Charniak, E. and Carroll, G. (1994). “Context-sensitive statistics for improved grammatical language models.” In *Proc. of the 12th National Conf. on Artificial Intelligence*, pp. 728–733.

Chi, Z. and Geman, S. (1998). “Estimation of probabilistic context-free grammars.” *Computational Linguistics*, **24** (2), 299–305.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). “Maximum likelihood from incomplete data via the EM algorithm.” *Journal of Royal Statistical Society*, **B39** (1),

1–39.

Fujisaki, T., Jenelik, F., Cocke, J., Black, E., and Nishino, T. (1989). "Probabilistic parsing method for sentence disambiguation." In *Proc. of the 1st Intl. Workshop on Parsing Technologies*, pp. 85–94.

池田思朗 (1995). "HMM の構造探索による音素モデルの生成." 電子情報学会論文誌 D-II, **J78-D-II** (1), 10–18.

Inui, K., Sornlertlemvanich, V., Tanaka, H., and Tokunaga, T. (1998). "Probabilistic GLR parsing: a new formalization and its impact on parsing performance." 自然言語処理, **5** (3), 33–52.

Kameya, Y. and Sato, T. (2000). "Efficient EM learning with tabulation for parameterized logic programs." In *Proc. of the 1st Intl. Conf. on Computational Logic*, Vol. 1861 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

北研二 (1999). 確率的言語モデル. 東京大学出版会.

Kita, K., Morimoto, T., Ohkura, K., Sagayama, S., and Yano, Y. (1994). "Spoken sentence recognition based on HMM-LR with hybrid language modeling." *IEICE Trans. on Information and Systems*, **E77-D** (2).

Kupiec, J. (1992). "An algorithm for estimating the parameters of unrestricted hidden stochastic context-free grammars." In *Proc. of Intl. Conf. on Computational Linguistics*, pp. 387–393.

Lafferty, J. D. (1993). "A Derivation of the Inside-Outside Algorithm from the EM algorithm." IBM research report, IBM T. J. Watson Research Center.

Lari, K. and Young, S. J. (1990). "The estimation of stochastic context-free grammars using the Inside-Outside algorithm." *Computer Speech and Language*, **4**, 35–56.

Magerman, D. and Marcus, M. (1991). "Pearl: a probabilistic chart parser." In *Proc. of the 5th Conf. on the European Chapter of the ACL*, pp. 15–20.

Magerman, D. and Weir, C. (1992). "Efficiency, robustness and accuracy in Picky chart parsing." In *Proc. of the 30th Conf. of the ACL*, pp. 40–47.

森高志, 亀谷由隆, 佐藤泰介 (2000). "確率文脈自由文法及びその拡張文法の高速 EM 学習法." 自然言語処理 139–12, pp. 85–92. 情報処理学会.

永田昌明 (1999). "形態素, 構文解析." 田中穂積 (編), 自然言語処理 —基礎と応用—, 1 章, pp. 2–45. 電子情報通信学会.

Pereira, F. and Schabes, Y. (1992). "Inside-Outside reestimation from partially bracketed corpora." In *Proc. of the 30th Annual Meeting of the ACL*, pp. 128–135.

Sato, T. (1995). "A statistical learning method for logic programs with distribution seman-

tics." In *Proc. of the 12th Intl. Conf. on Logic Programming*, pp. 715–729.

Sato, T. and Kameya, Y. (1997). "PRISM: A symbolic-statistical modeling language." In *Proc. of the 15th Intl. Joint Conf. on Artificial Intelligence*, pp. 1330–1335.

Stolcke, A. (1995). "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities." *Computational Linguistics*, **21** (2).

鷹見淳一 嵯峨山茂樹 (1993). "逐次状態分割法による隠れマルコフ網の自動生成." 電子情報学会論文誌 D-II, **J76-D-II** (10), 2155–2164.

Tamaki, H. and Sato, T. (1986). "OLD resolution with tabulation." In *Proc. of the 3rd Intl. Conf. on Logic Programming*, pp. 84–98.

田中穂積 (1988). 自然言語処理の基礎. 産業図書.

田中穂積, 竹澤寿幸, 衛藤純司 (1997). "MSLR 法を考慮した音声認識用日本語文法 — LR 表工学 (3) —." 音声言語情報処理 15–25, pp. 145–150. 情報処理学会.

Tomita, M. and Ng, S. (1991). "The generalized LR parsing algorithm." In Tomita, M. (Ed.), *Generalized LR Parsing*. Kluwer Academic Publishers.

付録

A　グラフィカル EM アルゴリズムの正当化

本節では Fujisaki らの方法，I-O アルゴリズム，gEM アルゴリズムに同じ初期パラメタを与えたとき，収束条件を同一にすればパラメタが同一の値に収束することを示す．これにより gEM アルゴリズムが正当化される．具体的には gEM アルゴリズムが計算する $\eta[A\to\zeta]$, すなわち規則 $(A\to\zeta)\in R$ の期待適用回数が Fujisaki らの方法，および I-O アルゴリズムで得られるものと一致することを示せばよい．また，節 2.5 で見たように，Fujisaki らの方法と I-O アルゴリズムが計算する $\eta[A\to\zeta]$ は一致するので，ここでは Fujisaki らの方法と gEM アルゴリズムで計算される $\eta[A\to\zeta]$ を調べれば十分である．

はじめに，我々は gEM アルゴリズムの計算は支持グラフの 1 つ目の特徴である支持グラフ Δ_ℓ の再帰的巡回を考える．そして，以下では τ の部分支持グラフの開始（終了）ノードを「τ の開始（終了）ノード」と呼ぶことにする．節 3.2 で述べたように，我々は巡回中に基本ノードに付与されている規則を集めることにする．まず，Δ_ℓ 中に出現する $A\to\zeta$ が付与された基本ノードの一つ v に注目する．v は τ の部分支持グラフに含まれているとし，v が属する局所パスを E とおく．その状況を図 20 に示す．そして，$S(0,n_\ell)$ の開始ノードから出発して v を通過するような再帰的巡回を全通り行ない，そこで集められた規則列の集合を $\psi(v,\mathbf{w}_\ell)$ とおく（明らかに $\psi(v,\mathbf{w}_\ell)\subseteq\psi(\mathbf{w}_\ell)$ である）．

τ の開始ノードから E に沿って τ の終了ノードに至る巡回によって得られる部分規則列を

$\mathbf{r}_1$ とおく．また O_ℓ の先頭である $S(0,n_\ell)$ の開始ノードから出発し，τ が付与された中間ノード u（図 20）に至る巡回，および u から $S(0,n_\ell)$ の終了ノードに至る巡回によって得られた部分規則列をそれぞれ $\mathbf{r}_0$, $\mathbf{r}_2$ とおく．そして，このような $\mathbf{r}_1$ すべてから成る集合を $\psi_{\rm in}(v,\mathbf{w}_\ell)$ とおき，可能な $\mathbf{r}_0,\mathbf{r}_2$ の組 $\langle\mathbf{r}_0,\mathbf{r}_2\rangle$ から成る集合を $\psi_{\rm out}(v,\mathbf{w}_\ell)$ とする．すると，先に定義した $\psi(v,\mathbf{w}_\ell)$ は $\psi_{\rm in}(v,\mathbf{w}_\ell)$ と $\psi_{\rm out}(v,\mathbf{w}_\ell)$ の直積と同一視できる．以上の定義と，PCFG における規則適用に関する独立性の仮定より，

$$\begin{aligned}\sum_{\mathbf{r}'\in\psi(v,\mathbf{w}_\ell)} P(\mathbf{r}') &= \sum_{\langle\mathbf{r}_0,\mathbf{r}_1,\mathbf{r}_2\rangle\in\psi(v,\mathbf{w}_\ell)} P(\mathbf{r}_0,\mathbf{r}_1,\mathbf{r}_2)\\ &= \sum_{\mathbf{r}_1\in\psi_{\rm in}(v,\mathbf{w}_\ell)} \sum_{\langle\mathbf{r}_0,\mathbf{r}_2\rangle\in\psi_{\rm out}(v,\mathbf{w}_\ell)} P(\mathbf{r}_1)P(\mathbf{r}_0,\mathbf{r}_2)\\ &= \Big(\sum_{\mathbf{r}_1\in\psi_{\rm in}(v,\mathbf{w}_\ell)} P(\mathbf{r}_1)\Big)\Big(\sum_{\langle\mathbf{r}_0,\mathbf{r}_2\rangle\in\psi_{\rm out}(v,\mathbf{w}_\ell)} P(\mathbf{r}_0,\mathbf{r}_2)\Big)\end{aligned}$$

が成り立つ．手続き *Get-Inside-Probs* における $\mathcal{P}$ と $\mathcal{R}$ の計算を再帰的に追えば，手続き終了時に $\mathcal{R}[\ell,\tau,E]=\sum_{\mathbf{r}_1\in\psi_{\rm in}(v,\mathbf{w}_\ell)} P(\mathbf{r}_1)$ となることは明らか．また，手続き *Get-Expectations* の $\mathcal{Q}$ の計算を追えば，$\mathcal{Q}[\ell,\tau]=\sum_{\langle\mathbf{r}_0,\mathbf{r}_2\rangle\in\psi_{\rm out}(v,\mathbf{w}_\ell)} P(\mathbf{r}_0,\mathbf{r}_2)$ であることが分かる．よって $\mathcal{Q}[\ell,\tau]\cdot\mathcal{R}[\ell,\tau,E]=\sum_{\mathbf{r}'\in\psi(v,\mathbf{w}_\ell)} P(\mathbf{r}')$ となる．これより *Get-Expectations* 行 10 で $\eta[A\to\zeta]$ に足し込まれる値は $\frac{1}{P(\mathbf{w}_\ell)}\sum_{\mathbf{r}'\in\psi(v,\mathbf{w}_\ell)} P(\mathbf{r}')$ に等しい．$A\to\zeta$ が付与された他の基本ノードについても同じ作業が行なわれ，さらにこれを $\ell=1\ldots N$ で繰り返すので，最終的に $\eta[A\to\zeta]$ は次のように計算される：

$$\eta[A\to\zeta]=\sum_{\ell=1}^{N}\frac{1}{P(\mathbf{w}_\ell)}\sum_{v:\ A\to\zeta\text{ is attached}}\ \sum_{\mathbf{r}'\in\psi(v,\mathbf{w}_\ell)} P(\mathbf{r}')\ . \tag{23}$$

ところで，$S(0,n_\ell)$ の開始ノードから出発する一つの再帰的巡回を考え，そこで集められた規則列を $\mathbf{r}$ とする．そのとき，この巡回において $A\to\zeta$ が付与された基本ノードを通過する回数は $\sigma(A\to\zeta,\mathbf{r})$ である．従ってこの $\mathbf{r}\in\psi(\mathbf{w}_\ell)$ について，和 $\sum_{v:\ A\to\zeta\text{ is attached}}\sum_{\mathbf{r}'\in\psi(v,\mathbf{w}_\ell)}$ では

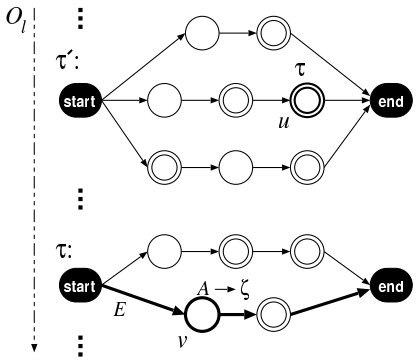

図 20　支持グラフ Δ_ℓ に出現する $A\to\zeta$ が付与された基本ノード v.

$\mathbf{r}$ が $\sigma(A\to\zeta,\mathbf{r})$ 回重複して数え上げられている．よって式 23 は Fujisaki らの計算式（式 14）と同値になる．以上と本節冒頭に述べた注意より，Fujisaki らの方法，I-O アルゴリズム，gEM アルゴリズムに同じ初期パラメタを与えたとき，収束条件を同一にすればパラメタは同じ値に収束する．

B　Stolcke の方法との対応

本節では Stolcke が提案した確率的 Earley パーザを用いる PCFG の訓練法 (Stolcke 1995) と提案手法において Earley パーザと gEM アルゴリズムを組み合わせた場合を簡単に対応づける．はじめに確率的 Earley パーザを簡単に記述する[17]．

B.1　確率的 Earley パーザ

Earley パーザは入力文 $\mathbf{w}_\ell$ の各単語位置をアイテム集合（Earley チャート）I_ℓ に基づいて構文解析を行なう．各アイテムは “$d' : {}_dA\to\zeta.\xi$” の形をしており，(i) 現在のポインタの位置が d'　($\mathbf{w}^{(\ell)}_{0,d'} = w^{(\ell)}_1\cdots w^{(\ell)}_{d'}$ が解析済み）であること，(ii) 非終端記号 A が統治する部分単語列が位置 d から始まること，(iii) A の展開は規則 $A\to\zeta\xi$ を用いて進められ，ドットの場所まで展開されていること，を表す．確率的 Earley パーザでは，Earley パーザに確率的な拡張が施されており，各アイテムに内側確率 $\beta_{d'}({}_dA\to\zeta.\xi)$ が式 24 のように付与される（式中の β は $\beta_{d'}({}_dA\to\zeta.\xi)$ の略記）[18]．内側確率 $\beta_{d'}({}_dA\to\zeta.\xi)$ はアイテム $d : {}_dA\to.\zeta\xi$ から始まり，$d' : {}_dA\to\zeta.\xi$ に至る経路の確率和である．

$$d' : {}_dA\to\zeta.\xi \;\; [\beta] \tag{24}$$

内側確率は次の 3 つの操作に従って計算される．

- ◇ **Prediction:** アイテム $d' : {}_dA\to\zeta.B\xi$ $[\beta]$ が存在し，かつ $(B\to\nu)\in R$ であるとき，アイテム $d' : {}_{d'}B\to.\nu$ $[\beta']$ が I_ℓ 中に存在しなければ，そのアイテムを I_ℓ に追加する．ただし，$\beta' = \theta(B\to\nu)$ である．もし存在すれば何も行なわない．
- ◇ **Scanning:** アイテム $(d'-1) : {}_dA\to\zeta.w^{(\ell)}_{d'}\xi$ $[\beta]$ が I_ℓ 中に存在するとき，$d' : {}_dA\to\zeta w^{(\ell)}_{d'}.\xi$ $[\beta']$ が I_ℓ に存在しなければ，これを I_ℓ に追加する．ただし $\beta' = \beta$ である．
- ◇ **Completion:** $d''<d'$ なる[19]2 つのアイテム $d' : {}_{d''}B\to\nu.$ $[\beta'']$ および $d'' : {}_dA\to\zeta.B\xi$ $[\beta]$ が I_ℓ に存在するとき，$d' : {}_dA\to\zeta B.\xi$ $[\beta']$ が I_ℓ に存在しなければ，これを I_ℓ に追加する．ただし，$\beta' = \beta\cdot\beta''$ である．存在しているときは $\beta' \mathrel{+}= \beta\cdot\beta''$ とする．

17 確率的 Earley パーザの記述は基本的に Stolcke の記法に従うが，本論文の記法に合わせた箇所もある．また，用語は (田中 1988) のものを用いる場合がある．

18 (Stolcke 1995) では内側確率の他に前向き確率と呼ばれる確率値を各アイテムに付与するが，EM 学習とは無関係なのでここでは省略する．その他にも EM 学習と無関係な記述は省略されている．

19 $d''\le d'$ と等号が入らないのは，我々が ε 規則をもたず $A\overset{+}{\Rightarrow}A$ とならない文法構造 (CFG) を仮定しているためである．Stolcke は両者を許しているので，彼の記述では等号が入っている．

EM 学習においては更にアイテム $d' :_d A \to \zeta.\xi$ の外側確率 $\alpha_{d'}({}_d A \to \zeta.\xi)$ を考慮する．この確率は (i) 初期アイテム $0 :_0 \to .S$ から出発し，(ii) $\mathbf{w}_{0,d}^{(\ell)}$ を生成し，(iii) ある ν について ${}_d A \to .\nu\xi$ を通り，(iv) ${}_{d'} A \to \nu.\xi$ から出発して $\mathbf{w}_{d',n_\ell}^{(\ell)}$ を生成し，(v) 最終アイテム $n_\ell :_0 \to S.$ で終わるような経路の確率和である．

◇ **Reverse completion:** I_ℓ 中の $d' :_{d''} B \to \nu.$ $[\alpha'', \beta'']$ と $d'' :_d A \to \zeta.B\xi$ $[\alpha', \alpha']$ の組すべてに対し，$d' :_d A \to \zeta B.\xi$ $[\alpha, \beta]$ を見つけ，$\alpha' \mathrel{+}= \beta'' \cdot \alpha$ と $\alpha'' \mathrel{+}= \beta' \cdot \alpha$ を行う．

ただし，アイテム $n_\ell :_0 S \to \nu.$ $[\alpha]$ のみ $\alpha := 1$ と初期化し，それ以外は $\alpha := 0$ としておく．すべての内側確率と外側確率を計算し終ったら，規則 $A \to \zeta$ の期待適用回数を次のように求め，後は I-O アルゴリズムの式 9 と同様にパラメタを更新する．

$$\eta[A \to \zeta] = \sum_{\ell=1}^{N} \frac{1}{\beta_{n_\ell}(0 \to S.)} \sum_{(d:{}_d A \to .\zeta) \in I_\ell} \alpha_d({}_d A \to .\zeta)\beta_d({}_d A \to .\zeta) \tag{25}$$

B.2　対応する支持グラフ

図 11 のメインルーチン *Learn-PCFG* における支持グラフ抽出ルーチン *Extract-CYK* を Earley パーザ用の支持グラフ抽出ルーチン *Extract-Earley* （図 21）に置き換える．そのサブルーチン *Visit-Earley* を図 22 に示す．gEM アルゴリズムは汎用であるため，手続きを特に変更する必要はない．*Extract-Earley* は Earley パーザの構文木出力ルーチンに基づいて記述されており，支持グラフ $\Delta_\ell = \langle O_\ell, \tilde{\psi}_\ell \rangle$ を生成する．$\tilde{\psi}_\ell$ は次のような形をしている．

$$\tilde{\psi}_\ell(d :_d B \to .\nu) = \{\{B \to \nu\}\} \tag{26}$$

$$\tilde{\psi}_\ell(d' :_d A \to \zeta w_{d'}^{(\ell)}.\xi) = \{\{(d'-1) :_d A \to \zeta.w_{d'}^{(\ell)}\xi\}\} \tag{27}$$

$$\tilde{\psi}_\ell(d' :_d A \to \zeta B.\xi) = \left\{ \{(d'' :_d A \to \zeta.B\xi),\ (d' :_{d''} B \to \nu.)\} \;\middle|\; d \le d'' < d', (d' :_{d''} B \to \nu.) \in I_\ell \right\} \tag{28}$$

式 26 の部分支持グラフに対する gEM アルゴリズムの内側確率の計算が確率的 Earley パー

```
1: procedure Extract-Earley() begin
2:     for ℓ := 1 to N do
3:         Initialize all ψ̃_ℓ(·) to ∅ and all Visited[·] to NO;
4:         ClearStack(U);
5:         Visit-Earley(ℓ, n_ℓ :_0 → S.);
6:         for k := 1 to |U| do τ_k := PopStack(U);
7:         O_ℓ := ⟨τ_1, τ_2, ..., τ_|U|⟩
8:     end
9: end.
```

図 21　Earley パーザ用の支持グラフ抽出ルーチン *Extract-Earley*().

ザの Prediction 操作に対応する．同様に式 27 に対する gEM アルゴリズムの内側確率計算が Scanning 操作に対応する．さらに，式 28 に対する gEM アルゴリズムの内側確率計算が Completion に，外側確率計算が Reverse completion 操作に各々対応する．そして，式 26 の部分支持グラフでの期待値計算が式 25 に対応する．

この場合の gEM アルゴリズムの計算量としては，式 28 の形の $\tilde{\psi}_\ell$ をもつ部分支持グラフに対する部分が効いてくる．Chomsky 標準形を満たす文法に対しては，このような部分支持グラフのノード数は可能な規則および単語位置 d, d', d'' の組合せ数 $O(|R|L^3)$ となる（R は規則集合，L はコーパス中の最大文長）．$R = R_{\max}$ のとき gEM アルゴリズムの最悪計算量は Stolcke の確率的 Earley パーザのものと同じく $O(|V_{\mathrm{n}}|^3 L^3)$ となる．また，Chomsky 標準形を満たさない場合を考え，規則右辺の最大記号数を m とおく．提案手法において，式 5 のような部分木の親子対を構成要素とする WFST をもつパーザ（GLR など）を用いた場合の計算量は $O(L^{m+1})$ となるが，Earley パーザを用いた場合の計算量は Stolcke の確率的 Earley パーザと同じく，m によらず $O(L^3)$ になる．ただ，GLR は LR 表への事前コンパイル・ボトムアップ計算等の好ましい性質を持っており，対象とする文法の特徴に応じてパーザを使い分けることが重要と思われる．

```
1: procedure Visit-Earley(ℓ, d' :_d A → ζ.ξ) begin
2:     Put τ = (d' :_d A → ζ.ξ)
3:     Visited[τ] := YES;
4:     if ζ = ε and d' = d then     /* d :_d A → .ξ (Prediction) */
5:         ψ̃_ℓ(τ) := {{A → ξ}}
6:     else if ζ = ζ' w_{d'}^{(ℓ)} then begin     /* d' :_d A → ζ' w_{d'}^{(ℓ)} .ξ (Scanning) */
7:         ψ̃_ℓ(τ) := {{(d' − 1) :_d A → ζ'.w_{d'}^{(ℓ)} ξ}};
8:         if Visited[(d' − 1) :_d A → ζ'.w_{d'}^{(ℓ)} ξ] = NO then Visit-Earley(ℓ, (d' − 1) :_d A → ζ'.w_{d'}^{(ℓ)} ξ)
9:     end
10:    else begin
11:        Put ζ = ζ' B;     /* B は非終端記号; d' :_d A → ζ' B.ξ (Completion) */
12:        foreach d'' such that d ≤ d'' < d' and (d'' :_d A → ζ'.Bξ), (d' :_{d''} B → ν.) ∈ I_ℓ do begin
13:            Add a set {(d'' :_d A → ζ'.Bξ), (d' :_{d''} B → ν.)} to ψ̃_ℓ(τ);
14:            if Visited[d'' :_d A → ζ'.Bξ] = NO then Visit-Earley(ℓ, d'' :_d A → ζ'.Bξ);
15:            if Visited[d' :_{d''} B → ν.] = NO then Visit-Earley(ℓ, d' :_{d''} B → ν.)
16:        end
17:    end;
18:    PushStack(τ, U)
19: end.
```

図 22　*Extract-Earley*() のサブルーチン *Visit-Earley*.

略歴

亀谷 由隆: 1995 年東京工業大学工学部情報工学科卒業．1997 年同大学大学院情報理工学研究科 修士課程修了．2000 年 9 月同研究科博士後期課程修了．博士（工学）．同研究科技術補佐員を経て，現在同研究科リサーチアソシエイト．論理プログラミングに基づく確率推論システム研究に従事．人工知能学会会員．

森 高志: 1999 年東京工業大学工学部情報工学科卒業．1999 年同大学大学院情報理工学研究科修士課程入学，現在在学中．

佐藤 泰介: 1973 年東京工業大学工学部電子物理学科卒業．1975 年同大学大学院修士課程修了．工学博士．同年，電子技術総合研究所入所．以来，人工知能の研究に従事．1995 年以来，東京工業大学大学院情報理工学研究科教授．人工知能学会，情報処理学会，EATCS 各会員．

(2000 年 5 月 10 日 受付)
(2000 年 10 月 10 日 採録)