■6 群（コンピュータ - 基礎理論とハードウェア） - 2 編（計算論とオートマトン）

3 章 文脈自由文法とプッシュダウンオートマトン

（執筆者：岩本宙造）［2010 年 3 月 受領］

■概要■

本章では，文脈自由文法，プッシュダウンオートマトン，構文解析，LL(k) 文法，文法の標準形について概覧する．まず，2 型句構造文法である文脈自由文法の定義を与え，その性質を解説するとともに，線形文法や文脈自由文法の拡張についても述べる．次に，プッシュダウンオートマトンを定義し，空スタック受理言語と最終状態受理言語の関係を述べる．また，決定性プッシュダウンオートマトンの部分クラスの性質を概説する．その後，構文解析について，導出と所属，様々な構文解析法について説明する．さらに，LL(k) 文法や LR(k) 文法を適用した構文解析について説明する．最後に，文脈自由文法の二つの重要な標準形として，チョムスキー標準形とグライバッハ標準形を紹介する．

【本章の構成】

本章は，文脈自由文法（3-1 節），プッシュダウンオートマトン（3-2 節），構文解析（3-3 節），LL(k) 文法と LR(k) 文法（3-4 節），文脈自由文法の標準形（3-5 節）の 5 節からなる．

■6 群 - 2 編 - 3 章

3-1 文脈自由文法

（執筆者：関　浩之）[2009 年 2 月 受領]

3-1-1 文脈自由文法

非負整数の集合を $\mathbb{N} = \{0, 1, 2, \cdots\}$ と書く．ε で空記号列（空系列）を表す．

4 項組 $G = (N, T, P, S)$ を**文脈自由文法**（context-free grammar: cfg）という．ここで，N は**非終端記号**（nonterminal symbol）と呼ばれる記号の有限集合，T は**終端記号**（terminal symbol）と呼ばれる記号の有限集合（ただし $N \cap T = \emptyset$），P は $A \to \alpha$（ただし $A \in N$, $\alpha \in (N \cup T)^*$）のかたちの生成規則（production rule）の有限集合，$S \in N$ は開始記号（start symbol）と呼ばれる非終端記号である．生成規則を単に規則（rule）ともいう．$A \to \varepsilon$ のかたちの規則を **ε-規則**という．

$G = (N, T, P, S)$ を任意の cfg とする．任意の規則 $A \to \alpha \in P$ と記号列 $\gamma, \delta \in (N \cup T)^*$ に対して，$\gamma A \delta \Rightarrow_G \gamma\alpha\delta$ と書き，規則 $A \to \alpha$ を適用して $\gamma A\delta$ から $\gamma\alpha\delta$ が得られるなどという．記号列 β_i $(0 \le i \le n)$ に対して $\beta_0 \Rightarrow_G \beta_1 \Rightarrow_G \cdots \Rightarrow_G \beta_n$ が成り立つとき，$\beta_0 \overset{*}{\Rightarrow}_G \beta_n$ と書き，β_0 から β_n への**導出**（derivation）という（$\overset{*}{\Rightarrow}_G$ は $\Rightarrow_G$ の反射推移閉包である）．以降，文脈から G が明らかな場合は，$\Rightarrow_G, \overset{*}{\Rightarrow}_G$ の G を省略し $\Rightarrow, \overset{*}{\Rightarrow}$ と書く．

$L(G) = \{w \in T^* | S \overset{*}{\Rightarrow}_G w\}$ を，G によって**生成される言語**という．ある cfg によって生成される言語を文脈自由言語（context-free language: cfl）と呼ぶ．$S \overset{*}{\Rightarrow} \beta \in (N \cup T)^*$ が成り立つとき，β を（G における）**文型**（sentential form）という．

例 1　(1) cfg $G_1 = (\{S\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S)$ において，S から終端記号列への任意の導出は，$S \to aSb$ を有限回適用した後，$S \to \varepsilon$ を適用して終わる．$S \to aSb$ を $n(\ge 0)$ 回適用したとき，$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \cdots \Rightarrow a^nSb^n \Rightarrow a^nb^n$ である．従って，$L(G_1) = \{a^nb^n \mid n \ge 0\}$.
(2) cfg $G_2 = (\{S, A\}, \{a, b, c, d\}, \{S \to aSd, S \to A, A \to bAc, A \to \varepsilon\}, S)$ において，S から終端記号列への任意の導出は，ある $m, n \ge 0$ に対して，

$$S \Rightarrow aSd \Rightarrow \cdots \Rightarrow a^mSd^m \Rightarrow a^mAd^m \Rightarrow a^mbAcd^m \Rightarrow \cdots \Rightarrow a^mb^nAc^nd^m \Rightarrow a^mb^nc^nd^m$$

と書ける．従って，$L(G_2) = \{a^mb^nc^nd^m \mid m, n \ge 0\}$. □

左辺の非終端記号が同一である複数の規則を，右辺を '|' で区切って，$S \to aSb \mid \varepsilon$ のようにまとめて書く．以降，英大文字は非終端記号，a, a_1, a_2, b, c などアルファベット前半の英小文字は終端記号，S は開始記号を表すものとする．この約束のもとでは，cfg を書くとき規則のみを列挙すれば十分である．

記号列 β に現れる記号数を $|\beta|$ で表す．また β における記号 a の出現回数を $|\beta|_a$ で表す．例えば，$|abac| = 4, |abac|_a = 2$ である．β に現れる記号を逆順に並べた記号列を β^R と表す．

例 2　次の言語はすべて cfl である．

$\{a^mb^mc^nd^n \mid m, n \ge 0\}$　　$\{ww^R \mid w \in \{a, b\}^*\}$　　$\{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$
$\{a^ib^jc^k \mid i \ne j$ または $j \ne k\}$　　$\{w \in \{a, b\}^* \mid$ どの $u \in \{a, b\}^*$ についても $w \ne uu\}$ □

$G = (N, T, P, S)$ を cfg とする．以下の条件を満たす任意の木を，（G における）導出木と

いう．
(1) 内部頂点のラベルは非終端記号，葉頂点のラベルは終端記号または ε，根頂点のラベルは開始記号 S である．
(2) ある頂点 v のラベルが非終端記号 A で，左から順にラベルが $X_1, X_2, \ldots, X_k$ であるような $k(\geq 1)$ 個の子頂点をもつならば，$A \to X_1X_2 \cdots X_k \in P$（各 $X_i \in (N \cup T)$）であるか，または，$k = 1, X_1 = \varepsilon$ かつ $A \to \varepsilon \in P$ である．

規則 $S \to AB, A \to aA \mid Ab \mid \varepsilon, B \to c$ で定まる cfg を考える．以下は，すべてこの文法における導出である．

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow abB \Rightarrow abc \tag{3·1}$$

$$S \Rightarrow AB \Rightarrow Ac \Rightarrow aAc \Rightarrow aAbc \Rightarrow abc \tag{3·2}$$

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAc \Rightarrow aAbc \Rightarrow abc \tag{3·3}$$

$$S \Rightarrow AB \Rightarrow AbB \Rightarrow aAbB \Rightarrow abB \Rightarrow abc \tag{3·4}$$

これらはすべて同じ終端記号列 abc を導出しており，式 (3·1)，式 (3·2)，式 (3·3) に対応する導出木はすべて等しい（図 3·1(a)）．式 (3·1) では導出途中の文型において常に一番左の非終端記号に規則を適用しているが，式 (3·2) では常に一番右の非終端記号に規則を適用している．前者のような導出を**最左導出**（leftmost derivation），後者のような導出を**最右導出**（rightmost derivation）と呼ぶ．もちろん式 (3·3) のように最左導出でも最右導出でもないような導出もある．一方，式 (3·1)，式 (3·2)，式 (3·3) に対応する導出木と式 (3·4) に対応する導出木は異なる（それぞれ図 3·1(a) と (b)）．ちなみに，式 (3·4) も最左導出である．この例のように，一つの導出木に複数個の導出が対応することがある．しかし，導出木と最左導出，導出木と最右導出は 1 対 1 に対応する．導出木はプログラミング言語や自然言語の構文構造を表現するのに用いられる，また最左導出及び最右導出は cfg の構文解析において重要な概念である〔本章 3-3, 3-4 参照〕．

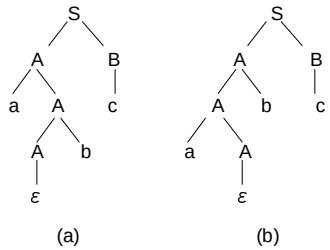

図 3·1　導出木

異なる導出木をもつような終端記号列が $L(G)$ に一つでも存在するとき，G は**あいまいである**という．$L = L(G)$ を満たす cfg G がすべてあいまいであるような cfl L を本質的にあいまいであるという．$\{a^m b^m c^n d^n \mid m, n \geq 0\} \cup \{a^m b^n c^n d^m \mid m, n \geq 0\}$ は本質的にあいまいな cfl の例である．

3-1-2 文脈自由言語の性質

(1) 周期性

L を cfl とするとき，十分長い任意の $z \in L$ に対し，$z = uvwxy$ とかけて，v, x を任意の回数繰り返した記号列 uv^iwx^iy もまた L に属するという性質（周期性）がある．

定理 1 (cfl のポンプ補題) L を cfl とする．次の条件を満たす正整数 n が存在する．

> $z \in L$ かつ $|z| \geq n$ ならば，$z = uvwxy$，ただし，$|vwx| \leq n, |vx| \geq 1$，とかけて，任意の $i \geq 0$ に対して，$uv^iwx^iy \in L$. □

ある言語が cfl であることを示すには，その言語を生成する cfg を具体的に与えればよい．一方，ポンプ補題を用いてある言語が cfl でないことを証明できることがある．

例 3 次の言語が cfl でないことをポンプ補題によって証明できる．

$\{a^nb^nc^n \mid n \geq 0\}$　　$\{a^mb^nc^md^n \mid m, n \geq 0\}$　　$\{a^{i^2} \mid i \geq 1\}$　　$\{a^i \mid i \text{ は素数 }\}$　　□

ほかに，ポンプ補題を強めた命題（Ogden の補題[2, 3]など）や交換補題[3]が知られている．

集合 S の n 回の直積を S^n とかく．終端記号集合 T の要素に適当に順番をつけて，$T = \{a_1, a_2, \ldots, a_n\}$ とする．任意の $w \in T^*$ に対し，$\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$ で定義される写像 $\Psi : T^* \to \mathbb{N}^n$ を，パリク（Parikh）写像という．例えば，$T = \{a_1, a_2, a_3\}$ に対し，$\Psi(a_2a_1a_2a_3) = (1, 2, 1)$. $\Psi(w)$ は，各記号の w における出現回数を表している．パリク写像を，$\Psi(L) = \{\Psi(w) \mid w \in L\}$ によって言語に対する写像に拡張する．例えば，$L_1 = \{a_1^na_2^{2n}a_3 \mid n \geq 0\}$ に対し，$\Psi(L_1) = \{(0, 0, 1), (1, 2, 1), (2, 4, 1), \ldots\}$. 任意の $\mathbf{b_0} \in \mathbb{N}^n$, 有限集合 $\mathbf{B} \subseteq \mathbb{N}^n$ に対し，$LN(\mathbf{b_0}, \mathbf{B}) = \{\mathbf{b_0} + \Sigma_{i=1}^k c_i\mathbf{b_i} \mid \mathbf{b_i} \in \mathbf{B}, c_i \in \mathbb{N}(1 \leq i \leq k)\}$ を線形集合と呼び，有限個の線形集合の和集合を準線形集合と呼ぶ．ここで，右辺の + 及び Σ はベクトルの加算である．

定理 2 (パリクの定理) 任意の cfl L に対し，$\Psi(L)$ は準線形集合である． □

例えば上の cfl L_1 に対し，$\Psi(L_1) = \{c(1, 2, 0) + (0, 0, 1) \mid c \in \mathbb{N}\} = LN((0, 0, 1), \{(1, 2, 0)\})$ と表すことができる（この場合，$\Psi(L_1)$ は線形集合である）．

(2) 閉包性

すべての cfl からなるクラス（言語の集まり）を $\mathcal{L}_{CF}$ と書く．ここでは，主な言語演算に対する $\mathcal{L}_{CF}$ の閉包性をまとめておく．

定理 3 $\mathcal{L}_{CF}$ は和集合, 連接, クリーン（Kleene）閉包，代入（従って準同型），正則言語〔2 章参照〕との積集合について閉じているが，積集合, 補集合については閉じていない．

(略証) 積集合について閉じていないこと（ほかは省略）．$\{a^mb^mc^n \mid m, n \geq 0\}, \{a^mb^nc^n \mid m, n \geq 0\}$ は cfl だがそれらの積集合 $\{a^nb^nc^n \mid n \geq 0\}$ は例 3 より cfl でない． □

定理 3 により，ある言語が cfl でないという既知の事実を用いて，ほかの言語が cfl でないことを証明できることがある．

例 4 $L = \{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c\}$ を考える．R を正則表現 $a^*b^*c^*$ で表される正則言語とすると，$L \cap R = \{a^nb^nc^n \mid n \geq 0\}$ である．もし，L が cfl であるとすると，定理 3 より，

$L \cap R$ も cfl である．これは例 3 に矛盾．従って，L は cfl ではない． □

(3) 判定問題

ここでは基本的な判定問題の判定可能性〔本編 4 章 4-4 参照〕をまとめておく．問題の入力としての cfl L は，$L = L(G)$ を満たす cfg G によって与えるものとする．また，終端記号列集合を T とする．

定理 4 (空集合，有限性，所属問題) L を任意に与えられた cfl とする．L が空集合かどうか，L が有限集合かどうか，与えられた $w \in T^*$ に対して $w \in L$ かどうかを判定する問題はいずれも判定可能である． □

なお所属問題は時間計算量 $O(|w|^3)$ で解け，一般の cfg の構文解析問題と関係が深い〔本章 3-3 参照〕．次は，正則言語の場合〔本編 2 章参照〕と異なる結果である．

定理 5 (全体集合，等価性，包含性問題) L_1, L_2 を任意に与えられた cfl とする．$L_1 = T^*$ か（L_1 は全体集合か），$L_1 = L_2$ か，$L_1 \subseteq L_2$ かを判定する問題はいずれも判定不能である． □

(4) 表現定理

$n+2$ 個の規則 $S \to SS \mid \varepsilon, S \to \langle_i S \rangle_i$ $(1 \le i \le n)$ からなる cfg によって生成される cfl D_n を（n 次の）Dyck 言語と呼ぶ．D_n は，正しく対応のとれた n 種類の括弧 $\langle_1, \rangle_1, \ldots, \langle_n, \rangle_n$ の列全体からなる集合である．任意の cfl は，ある Dyck 言語，正則言語，準同型によって以下のように表現できる．

定理 6 (Chomsky-Schüzenberger の定理) 任意の cfl L に対してある $n \ge 1$ と正則言語 R，準同型 h が存在して，$L = h(R \cap D_n)$ と表現することができる． □

3-1-3 線形文法

どの規則の右辺にも非終端記号がたかだか 1 回しか現れないような cfg を線形文法（linear cfg: lcfg）と呼び，lcfg によって生成される cfl を線形言語（lcfl）と呼ぶ．例えば，$\{a^n b^n \mid n \ge 0\}$ は，lcfg $S \to aSb \mid \varepsilon$ によって生成されるので lcfl である．lcfl にもポンプ補題が知られており，それを用いて，$\{a^m b^m c^n d^n \mid n \ge 0\}$ が lcfl ではないことなどが示せる．すべての正則言語，線形言語，文脈規定言語からなるクラスを順に，$\mathcal{L}_R, \mathcal{L}_{LIN}, \mathcal{L}_{CS}$ と書く．

定理 7 $\mathcal{L}_R \subseteq \mathcal{L}_{LIN} \subseteq \mathcal{L}_{CF} \subseteq \mathcal{L}_{CS}$．すなわち，正則言語，線形言語，文脈自由言語，文脈規定言語のクラスはこの順にすぐ右のクラスに真に含まれる．

（証明）正則文法は線形文法であるから，$\mathcal{L}_R \subseteq \mathcal{L}_{LIN}$．$\mathcal{L}_{LIN} \subseteq \mathcal{L}_{CF} \subseteq \mathcal{L}_{CS}$ は文法のクラスの定義より明らか．一方，$\{a^n b^n \mid n \ge 0\} \in \mathcal{L}_{LIN} \cap \overline{\mathcal{L}_R}$ かつ，$\{a^m b^m c^n d^n \mid n \ge 0\} \in \mathcal{L}_{CF} \cap \overline{\mathcal{L}_{LIN}}$．また，例えば例 3 の最初の言語は $\mathcal{L}_{CS} \cap \overline{\mathcal{L}_{CF}}$ に属する． □

3-1-4 cfg の拡張

cfg の拡張は種々行われているが，ここでは cfg の自然な拡張であり，その生成能力が cfg より真に大きく csg より真に小さい 3 種の文法の概略を述べる（解説書として，文献 4) 2 巻

3 章，3 巻 1-3 章を参照されたい）．

接木文法（tree adjoining grammar: tag）では規則右辺は木で与えられ，導出において，導出木の葉頂点だけでなく内部頂点を規則右辺の木で置き換えることができる．

インデックス文法（indexed grammar: ig）では導出において，非終端記号にインデックスと呼ばれる記号からなる列を対応づけ，文型の書き換えと同時にインデックス列の書換えも行う．またインデックス列の内容によって規則の適用を制御できる．マクロ文法（macro grammar），文脈木文法（context-free tree grammar）もインデックス文法とその生成する言語クラスが等しいことが知られている．

多重文脈自由文法（multiple cfg: mcfg）は，非終端記号から終端記号列の組を生成できるように cfg を拡張したものである．mcfg と生成能力の等しい文法として，linear context-free rewriting system, finite copying tree-to-string transducer, local unordered scattered context grammar などがある．

tag, ig, mcfg の生成する言語クラスを順に $\mathcal{L}_{TA}, \mathcal{L}_{IX}, \mathcal{L}_{MCL}$ と書く．包含関係 $\mathcal{L}_{CF} \subsetneq \mathcal{L}_{TA} \subsetneq \mathcal{L}_{MCL} \subsetneq \mathcal{L}_{CS}$ かつ $\mathcal{L}_{TA} \subsetneq \mathcal{L}_{IX} \subsetneq \mathcal{L}_{CS}$ が知られている．$\mathcal{L}_{IX}$,$\mathcal{L}_{MCL}$ は，$\mathcal{L}_{CF}$ の言語演算に対する閉包性，判定問題の判定可能性の多くを保存している．一つの大きな差異は，mcfg の所属問題が入力記号列長の多項式時間で解けるのに対し，ig にはそのようなアルゴリズムは知られていないことである．また，$\mathcal{L}_{IX} \cap \overline{\mathcal{L}_{MCF}} \neq \emptyset$ であるが，$\mathcal{L}_{MCF} \subsetneq \mathcal{L}_{IX}$ かどうかは未解決である．

3-1-5 おわりに

cfg についてより詳しく学びたい読者は，文献 2, 3) などを読まれるとよい．文献 5) は簡潔にまとめられている．和書にも参考文献にあげたような良書が数多くある．cfg はプログラミング言語や自然言語の構文解析に広く応用されてきた〔本章 3-3, 3-4 参照〕．近年，cfg の生物系列解析への応用が注目されている．興味のある読者は文献 1, 10) を参照されたい．

■参考文献

1) R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, “Biological Sequence Analysis - Probabilistic Models of Proteins and Nucleic Acids,” Chap.9 and 10, Cambridge University Press, 1998.
2) J.E. Hopcroft and J.D. Ullman, “Introduction to Automata Theory, Languages, and Computation,” Addison Wesley, 1979; (野崎昭弘, 高橋正子, 町田元, 山崎秀記訳, “オートマトン 言語理論 計算論 I,” サイエンス社, 1984); 2nd ed., J.E. Hopcroft, R. Motowani and J.D. Ullman, 2001.
3) J. Berstel and L. Boasson, “Context-free languages,” in Handbook of Theoretical Computer Science, Vol.B, ed. by J. van Leeuwen, pp.59-102, The MIT Press, 1990.
4) “Handbook of Formal Languages,” ed. by G. Rozenberg and A. Salomma, Springer, 1997.
5) M. Sipser, “Introduction to the Theory of Computation,” PWS Publishing, 1997
6) 岩間一雄, “オートマトン・言語と計算理論,” コロナ社, 2003.
7) 都倉信樹, “オートマトンと形式言語,” 昭晃堂, 1995.
8) 富田悦次, 横森貴, “オートマトン・言語理論,” 森北出版, 1992.
9) 丸岡章, “計算理論とオートマトン言語理論,” サイエンス社, 2005.
10) 丸山修, 阿久津達也, “バイオインフォマティクス - 配列データ解析と構造予測 -,” 朝倉書店, 2007.
11) 守屋悦朗, “形式言語とオートマトン,” サイエンス社, 2001.
12) 米田政明, 広瀬貞樹, 大里延康, 大川知, “オートマトン・言語理論の基礎,” 近代科学社, 2003.

■6 群 - 2 編 - 3 章

3-2 プッシュダウンオートマトン

（執筆者：関　浩之）[2009 年 2 月 受領]

3-2-1 プッシュダウンオートマトン

記号や用語は本章 3-1 で定義したものを用いる．記号列 u, v, w に対し $w = uv$ のとき，u を w の接頭辞，v を w の接尾辞という．

6 項組 $M = (Q, T, \Gamma, R, q_0, Q_F)$ を**プッシュダウンオートマトン**（pushdown automaton: pda）という．ここで，Q は**状態**の有限集合，T は**入力記号**の有限集合，Γ は**スタック記号**の有限集合，R は $(Q \times (T \cup \{\varepsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$ の有限部分集合，$q_0 \in Q$ は**初期状態**，$Q_F \subseteq Q$ は**最終状態**の集合である．Γ にはボトム記号とよばれる特別な記号 \$ が属する．R は遷移関係と呼ばれる．$(q, a, X, p, \gamma) \in R$ を $(q, a, X) \to_R (p, \gamma)$ とかき，**遷移規則**（または単に規則）と呼ぶ．特に $a = \varepsilon$ のとき **ε-（遷移）規則**という．

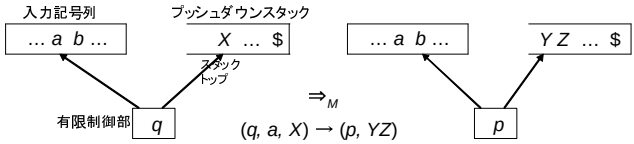

図 3･2　pda **の遷移例**

pda は図 3･2 に示すように，状態 $q \in Q$ を記憶する有限制御部，入力記号列 $w \in T^*$（入力ヘッドによって左から順に 1 記号ずつ読むことができる）及びプッシュダウンスタック（スタックと略）からなる．pda は有限制御部が初期状態 q_0 であり，入力記号列の最左端を入力ヘッドが指し，スタックがボトム記号 \$ のみからなる状況から，遷移関係 R に従って動作する．規則 $(q, a, X) \to_R (p, \gamma)$ は，「状態が q で入力ヘッドの指す記号が a であり，スタックトップの記号（図ではスタックの左端）が X ならば，状態を p に変更し X を γ に置き換え，$a \neq \varepsilon$ ならば入力ヘッドを一つ右に動かし，$a = \varepsilon$ ならば入力ヘッドは動かさない」ことを表す．状態が q, 入力ヘッド以降の入力記号列が $w \in T^*$, スタックの内容が $\beta \in \Gamma^*$ である状況を (q, w, β) と書く．任意の $(q, a, X) \to_R (p, \gamma), u \in T^*, \alpha \in \Gamma^*$ に対し，$(q, au, X\alpha) \Rightarrow_M (p, u, \gamma\alpha)$ と書き，$\Rightarrow_M$ の反射推移閉包を $\overset{*}{\Rightarrow}_M$ と書く．文脈から R や M が明らかなときは，$\to_R, \Rightarrow_M, \overset{*}{\Rightarrow}_M$ を $\to, \Rightarrow, \overset{*}{\Rightarrow}$ と書く．$(q, a, X) \to_R (p, \gamma)$ は，$|\gamma| = 0$（つまり $\gamma = \varepsilon$）のときポップ規則，$|\gamma| \geq 2$ のときプッシュ規則と呼ぶ．

例 5　pda $M_1 = (\{q_0, q_1\}, \{a, b\}, \{A, \$\}, R_1, q_0, \{q_1\})$ を考える．ここで R_1 に属する規則は，

$$(q_0, a, \$) \to (q_0, A\$) \quad (q_0, a, A) \to (q_0, AA)$$
$$(q_0, b, A) \to (q_1, \varepsilon) \quad (q_1, b, A) \to (q_1, \varepsilon) \quad (q_1, \varepsilon, \$) \to (q_1, \varepsilon)$$

M_1 は状態 q_0 で a を読んだら A をスタックにプッシュし状態は q_0 のままとする．また状態 q_1, q_2 において b を読んだら A をポップし状態を q_1 とする．更にスタックトップがボトム記号 \$ のとき ε-規則により \$ をポップする．よって，入力が $a^n b^n$（$n \geq 1$）のときかつそのとき

に限り，入力を読みきってスタックを空にできる．例えば，$(q_0, aabb, \$) \Rightarrow_{M_1} (q_0, abb, A\$) \Rightarrow_{M_1} (q_0, bb, AA\$) \Rightarrow_{M_1} (q_1, b, A\$) \Rightarrow_{M_1} (q_1, \varepsilon, \$) \Rightarrow_{M_1} (q_1, \varepsilon, \varepsilon)$.

pda $M_2 = (\{q_0, q_1\}, \{a, b\}, \{A, B, \$\}, R_2, q_0, \emptyset)$ を考える．ここで R_2 に属する規則は，

$$(q_0, a, X) \to (q_0, AX) \quad (q_0, b, X) \to (q_0, BX) \quad (q_0, \varepsilon, X) \to (q_1, X) \quad (X \in \{\$, A, B\})$$
$$(q_1, a, A) \to (q_1, \varepsilon) \qquad (q_1, b, B) \to (q_1, \varepsilon) \qquad (q_1, \varepsilon, \$) \to (q_1, \varepsilon)$$

M_2 は状態 q_0 で a, b を読んだらそれぞれ A, B をプッシュする．また，q_0 から ε-規則によりスタックを変化させずに状態を q_1 に変えられる．q_1 でスタックトップが A, B のときそれぞれ a, b を読んだらポップする．したがって，入力が ww^R（$w \in \{a, b\}^*$）であるときかつそのときのみ，入力をすべて読みきってスタックを空にできる．ε-規則は状態 q_0 のときいつでも適用できる．入力記号列をちょうど半分読んだ直後に ε-規則を適用したときに限り入力を読みきって空スタックにできることに注意．例えば，$(q_0, abba, \$) \Rightarrow_{M_2} (q_0, bba, A\$) \Rightarrow_{M_2} (q_0, ba, BA\$) \Rightarrow_{M_2} (q_1, ba, BA\$) \Rightarrow_{M_1} (q_1, a, A\$) \Rightarrow_{M_2} (q_1, \varepsilon, \$) \Rightarrow_{M_2} (q_1, \varepsilon, \varepsilon)$. □

pda $M = (Q, T, \Gamma, R, q_0, Q_F)$ に対し空スタック受理言語 $N(M)$, 最終状態受理言語 $L(M)$ を

$$N(M) = \{w \in T^* \mid (q_0, w, \$) \overset{*}{\Rightarrow}_M (q, \varepsilon, \varepsilon), \exists q \in Q\}$$
$$L(M) = \{w \in T^* \mid (q_0, w, \$) \overset{*}{\Rightarrow}_M (q_f, \varepsilon, \beta), \exists q_f \in Q_F, \exists \beta \in \Gamma^*\}$$

と定義する．例 5 の M_1, M_2 に対して，$N(M_1) = \{a^n b^n \mid n \geq 1\}$, $N(M_2) = \{ww^R \mid w \in \{a, b\}^*\}$, $L(M_1) = \{a^m b^n \mid m \geq n \geq 1\}$, $L(M_2) = \emptyset$ である．このように，一般に $N(M) = L(M)$ は成り立たないが，次の性質が成り立つ．

定理 8 M を任意の pda とする．$N(M_1) = L(M)$, $L(M_2) = N(M)$ を満たす pda M_1, M_2 を M から構成できる．

（略証）M_1 は M を模倣し M が最終状態に入ったらスタックに残っている記号をすべてポップすればよい．M_2 は自分固有のボトム記号 \$ の上に M のボトム記号（記号名を変える）を積んでから M を模倣し，スタックが \$ だけになったら最終状態に入ればよい． □

定理 9 任意の cfg G に対し，$N(M) = L(G)$ を満たす pda M を G から構成できる．逆に任意の pda M に対し，$L(G) = N(M)$ を満たす cfg G を M から構成できる．

（略証）$G = (N, T, P, S)$ を任意の cfg とする．M は $N \cup T$ をスタック記号集合，S をボトム記号とし，G の最左導出を模倣する．スタック上には導出の各ステップの文型（の接尾辞）を記憶し，入力記号とスタックトップが一致したらその記号をポップする．$w \in L(G)$ であるときかつそのときのみ，入力 w に対して M は G における w の最左導出を模倣して空スタックで終わることができる．逆に，$M = (Q, T, \Gamma, R, q_0, Q_F)$ を任意の pda とする．M を模倣するための cfg G は $X[q, p]$ $(q, p \in Q, X \in \Gamma)$ のかたちの非終端記号をもち，

$(q, w, X) \overset{*}{\Rightarrow}_M (p, \varepsilon, \varepsilon)$ であるときかつそのときのみ $X[q, p] \overset{*}{\Rightarrow}_G w$

が成り立つように規則を構成する．例えば，M にプッシュ規則 $(q, a, X) \to (p, YZ)$ があれば，規則群 $X[q, q_2] \to Y[p, q_1]\, Z[q_1, q_2]$ $(\forall q_1, q_2 \in Q)$ を G に含める． □

3-2-2 決定性 pda

$M = (Q, T, \Gamma, R, q_0, Q_F)$ を pda とする．次の条件 (1)(2) がともに成り立つとき，M を決定

性 pda（dpda）と呼ぶ．(1) 各 $q \in Q, a \in T \cup \{\varepsilon\}, X \in \Gamma$ について，左辺が (q, a, X) の規則はたかだか一つしか R に属さない．(2) $q \in Q, X \in \Gamma$ とする．左辺が (q, ε, X) の規則が R に属すなら左辺が (q, a, X) $(a \neq \varepsilon)$ の規則は R に属さない．

例 5 の M_1 は dpda だが M_2 は dpda ではない．dpda についても定理 8 に対応する性質が成り立つ．ある dpda に対して $L = L(M)$ となる言語を決定性 cfl（dcfl）と呼び，dcfl 全体からなるクラスを $\mathcal{L}_{DCF}$ と書く．

定理 10 $\mathcal{L}_{DCF}$ は補集合，正則言語との積集合について閉じている．$\mathcal{L}_{DCF}$ は和集合，連接，クリーン閉包，積集合，準同型（したがって代入）について閉じていない． □

cfl の等価性問題は判定不能である〔本章 3-1 参照〕．dcfl の等価性問題が判定可能であるかどうかは長期間未解決であったが，Sénizergues によって肯定的に解決された[3, 4]．

定理 11 (等価性問題) 与えられた dcfl L_1, L_2 に対し，$L_1 = L_2$ かどうかは判定可能である． □

3-2-3 そのほかの部分クラス

ここでは pda のほかの部分クラスを二つ紹介する．dpda M が次の条件をともに満たすとき，M を superdeterministic pda（sdpda）と呼ぶ[2]．(1) ε-規則による連続する遷移列の長さは M で定まるある定数以下となる．(2) $(q, w, \alpha_1) \overset{*}{\Rightarrow}_M (p_1, \varepsilon, \beta_1), (q, w, \alpha_2) \overset{*}{\Rightarrow}_M (p_2, \varepsilon, \beta_2)$ ならば，$p_1 = p_2, |\alpha_1| - |\beta_1| = |\alpha_2| - |\beta_2|$．すなわち，同一の状態から同一の入力に対して動作したら，到達する状態は同一でありスタック長の変化量も等しい．

ある sdpda $M = (Q, T, \Gamma, R, q_0, Q_F)$ が存在して $L = L(M) = \{w \in T^* \mid (q_0, w, \$) \overset{*}{\Rightarrow}_M (q_f, \varepsilon, \varepsilon), \exists q_f \in Q_F\}$ となる言語 L を，M によって受理される sdpdl と呼ぶ．与えられた cfl L_1 と sdpdl L_2 に対し，$L_1 \subseteq L_2$ かどうかは 2 重指数時間で判定可能である．

visibly pda (vpda)[1]では，入力記号集合は $T = T_c \cup T_r \cup T_i$ と分割され，T_c, T_r, T_i に属する入力記号に対して，それぞれ，プッシュ規則，ポップ規則，スタックを変化させない規則のみが定義される．プッシュ規則による遷移はスタックトップ記号に依存してはならない．正則言語のもつ性質の多くが vpda の受理言語 vpdl についても成り立っている: 任意の vpda を受理言語の等しい決定性 vpda へ変換できる．vpdl のクラスは和集合，積集合，補集合について閉じている．与えられた vpdl L_1, L_2 に対し，$L_1 \subseteq L_2$ かどうかは指数時間で判定可能である．

（本節に関する解説的な参考文献は前節と共通であるので省略する．）

■参考文献

1) R. Alur and P. Madhusudan, “Visibly pushdown languages,” Proc. of the 36th ACM Symposium on Theory of Computing (STOC’04), pp.202-211, 2004.
2) S. Greibach and E.P. Friedman, “Superdeterministic PDAs: A subcase with a decidable inclusion problem,” J. Assoc. Comput. Mach., vol.27, no.4, pp.675-700, 1980.
3) G. Sénizergues, “L(A)=L(B)? Decidability results from complete formal systems,” Theoret. Comput. Sci., vol.251, nos.1-2, pp.1-166, 2001.
4) G. Sénizergues: “L(A)=L(B)? A simplified decidability proof,” Theoret. Comput. Sci., vol.281, nos.1-2, pp.555-608, 2002.

■6 群 - 2 編 - 3 章

3-3 構文解析

（執筆者：椎名広光）[2008 年 12 月 受領]

3-3-1 構文解析の役割

構文解析の役割としては，入力文字列から文法規則の適用順を求め，適用順から文法規則の文法記号間の親子関係を木構造で表した構文解析木を作成する．別な観点からは，入力文字列の文法から生成する言語への所属を調べる役割がある．所属の問題は，構文解析の処理で文法に適合する構文解析木が作成できるとき，入力文字列が所属しているとすることができ，そうでなければ所属してないとしてよい．

3-3-2 導出と所属について

文法 $G = (N, T, P, S)$, N：非終端記号, T：終端記号, P：文法規則, S：開始記号, に対して，文法例 G_1=(N_1, T_1, P_1, S_1)=($\{S, A, B, C\}, \{a, b, c\}, \{S \to AC, A \to BA, A \to BC, A \to DC, D \to BA, A \to a, B \to b, C \to c\}, S$) とする． これに対して，入力文字列 *bbaac* の構文解析木が 2 種類作成でき，図 3･3(a), (b) に示す．

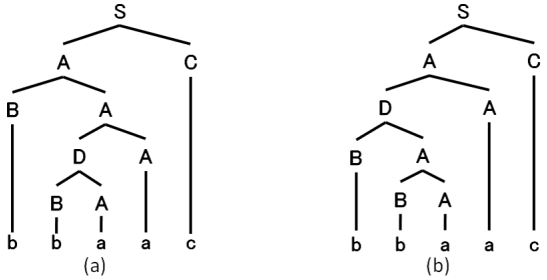

図 3･3　構文解析木の例

図 3･3(a), (b) は，開始記号 S から入力記号 *bbaac* の導出過程の例を，1 回の文法規則の適用による導出を $\Rightarrow$ で表すものとすると，次の (a), (b) のようになる．このように文法例 G_1 は同一の入力で複数の構文解析木が作成できるので，あいまいな文法という．なお，導出中の記号の最も左側の記号を文法規則で書き換えることを最左導出，記号の最も右側の記号を文法規則で書き換えることを最右導出と呼び，(a) は最左導出，(b) は最右導出の例である．

(a) $S \Rightarrow AC \Rightarrow BAC \Rightarrow bAC \Rightarrow bDAC \Rightarrow bBAAC \Rightarrow bbAAC \Rightarrow bbaAC \Rightarrow bbaaC \Rightarrow bbaaac$
(b) $S \Rightarrow AC \Rightarrow Ac \Rightarrow DAc \Rightarrow Dac \Rightarrow BAac \Rightarrow BBAac \Rightarrow BBaac \Rightarrow Bbaac \Rightarrow bbaaac$

また，0 回数以上の導出を $\Rightarrow^*$ とすると，S から入力記号列 *bbaac* への途中の導出過程を省略して $S \Rightarrow^* bbaac$ と書くことができる．文法 G に対応する言語 $L(G)(= \{w | S \Rightarrow^* w\})$ と定義することができ，言い換えればいずれか一つの構文解析木が作れれば，入力文字列がその文法から生成される言語に所属することになる．

なお，1 回の最左導出を $\Rightarrow_{lm}$ を，任意の回数の最左導出を $\Rightarrow^*_{lm}$ で表す．また，1 回の最右導出を $\Rightarrow_{rm}$ を，任意の回数の最右導出を $\Rightarrow^*_{rm}$ で表す．

3-3-3 構文解析木の作成

前節の入力文字列の所属では開始記号 S から入力文字列が導出したが，コンパイラや自然言語処理で構文解析の主な目的は入力記号から構文解析木を作成することである．しかし，図 3･3 のように構文解析木は複数生成することができる．そのためコンパイラの構文解析法では，どれか一つを作成する．どの一つの構文解析木を作成するかは，構文解析手法によって違い，その特徴は最右導出や最左導出の導出の種類によって区別される．また，構文解析の処理は，構文解析木の部分を確定していく方向によっても分けられる．構文解析木の開始記号から入力文字列に向けて木を確定していく下向き（図 3･4(a)），その反対に入力文字列側から開始記号に向けて木を確定していく上向き（図 3･4(b)），その混合型の三つに分けられる．LL(k) 文法は下向き，LR(k) 文法は上向き，LC(k) 文法は混合型の代表例である．

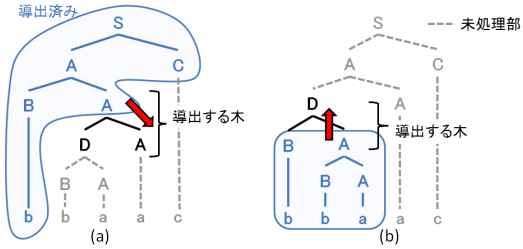

図 3･4　構文解析の処理

3-3-4 様々な構文解析法

構文解析法は，LL(k) 文法，LR(k) 文法に対する LL(k) 構文解析法，LR(k) 構文解析法のほかにも考案されている．LC(k) 文法や順位文法などが知られている．これらは文脈自由文法のより小さいクラスの文法で，コンパイラなどで利用される．

そのほか自然言語処理でも用いられる構文解析法として，CKY 法，Earley 法，Chart 法，GLR 法などが知られている．これらは，あいまい性を含む文脈自由文法を対象としており複数の構文解析木を作成することを考慮した処理となっている．

また，構文解析の処理時間は，文法の種類によって分けられる．LL(k) 文法，LR(k) 文法，LC(k) 文法のようなあいまいでない文法の場合では，入力文字列数を n とすると，構文解析木の作成に後戻りがなく逐次処理可能なので，O(n) 時間で処理できる．一方，あいまい性を含む文脈自由文法に対する構文解析の処理時間は，後戻りを考慮するため O(n^3) 時間であることが知られている[1)]．

■参考文献

1)　J. ホップフロフト, J. ウルマン, “オートマトン 言語理論 計算理論 I,II,” サイエンス社, 1986.

■6 群 - 2 編 - 3 章

3-4 LL(k) 文法と LR(k) 文法

（執筆者：椎名広光）[2008 年 12 月 受領]

3-4-1 LL(k) 文法

LL(k) 文法は左から右へ，最左導出の構文解析木を作る．ここで任意の記号列 γ と整数 $k \geq 0$ に対して，$head_k(\gamma)$ を次のように定義する．

$$head_k(\gamma) = \begin{cases} \zeta, \text{ただし} \gamma = \zeta\eta, |\zeta| = k, |\eta| \geq 0 \text{ の場合}, \\ \gamma, \text{ただし } |\gamma| < k \text{ の場合}. \end{cases}$$

[LL(k) 文法の定義[2)]**]** 文脈自由文法 $G = (N, T, P, S)$ が $S \Rightarrow_{lm}^{*} uA\nu$ で $A \to \alpha, A \to \beta \in P$, $\alpha \neq \beta$ であるとする． このとき，$S \Rightarrow_{lm}^{*} uA\nu \Rightarrow_{lm} u\alpha\nu \Rightarrow_{lm}^{*} ut_1$, $S \Rightarrow_{lm}^{*} uA\nu \Rightarrow_{lm} u\beta\nu \Rightarrow_{lm}^{*} ut_2$ に対して $head_k(t_1) \neq head_k(t_2)$ であるならば，文法 G を LL(k) 文法という（図 3･5(a)）．

LL(k) 文法に対応する LL(k) 構文解析において，途中までの入力文字列から適用する文法規則を決定するときに，これまでの入力文字列だけでなくこれから先に処理する入力文字列を利用する．これを先読みと呼び，k は利用する文字数である．ここで LL(2) 文法の文法例 $G_2=(N_2, T_2, P_2, S_2)=(\{S, A, B, C\}, \{a, b\}, \{S \to AA, S \to C, C \to BCB, C \to BAB, A \to a, B \to b\}, S)$ に対して，入力文字列 $bbabb$ の 1 文字目の b を処理し，2 文字目の b で C に対して $C \to BAB$ または $C \to BCB$ が適用される可能性があるが，先読み文字列が ba であることから $C \to BAB$ が唯一となる．この状況を図 3･5(b) に示す．

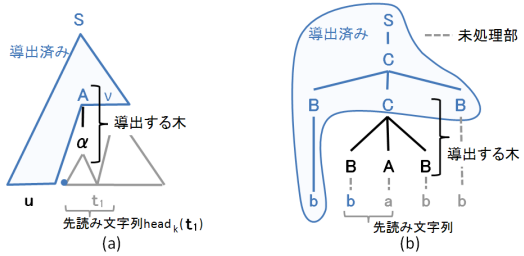

図 3･5　LL(k) 文法の定義と LL(2) 文法の文法規則決定過程

また，LL(k) 文法をそのまま構文解析の処理に適用すると文法規則を決めるのに使う解析表が大きいため，縮小した強 LL(k) 文法が知られている．

[強 LL(k) 文法の定義] 文脈自由文法 $G = (N, T, P, S)$ が $S \Rightarrow_{lm}^{*} uA\nu$ で $A \to \alpha, A \to \beta \in P$, $\alpha \neq \beta$ であるとする． このとき，$S \Rightarrow_{lm}^{*} uA\nu \Rightarrow_{lm} u\alpha\nu \Rightarrow_{lm}^{*} ut_1$, $S \Rightarrow_{lm}^{*} u'A\nu' \Rightarrow_{lm} u'\beta\nu' \Rightarrow_{lm}^{*} u't_2$ に対して $head_k(t_1) \neq head_k(t_2)$ であるならば，文法 G を強 LL(k) 文法という．

なお，先読み文字列数 $k = 1$ においては強 LL(1) 文法と LL(1) 文法は等価であるが，$k > 1$ においては強 LL(k) 文法は LL(k) 文法の真部分クラスである．

3-4-2 LR(k) 文法

LR(k) 文法は左から右へ，最右導出の構文解析木を作る．

[LR(k) 文法の定義[2]**]** 文脈自由文法 $G=(N,T,P,S)$ が $S \Rightarrow^*_{\mathrm{rm}} \mu Av \Rightarrow_{\mathrm{rm}} \mu\alpha v$, $S \Rightarrow^*_{\mathrm{rm}} \zeta Bv_1 \Rightarrow_{\mathrm{rm}} \zeta\beta v_1 = \mu\alpha v_2$ に対して，$head_k(v) = head_k(v_2)$ であるならば，$\zeta\beta v_1 = \mu Av_2$ となる文法 G を LR(k) 文法という（図 3･6(a)）．

前節の文法例 G_2 を LR(k) 文法に当てはめると LR(0) 文法となる．ここで $k=0$ で先読み文字列が 0 個で，文法規則の導出の決定に先読み文字列を使用せず適用する文法規則を決定できる．また，LR(0) 文法は決定性プッシュダウンオートマトンで処理ができることが知られている．図 3･6(b) は入力文字列 *bbabb* に対して，前節と同じ $C \to BAB$ を適用する順序は，4 文字目の入力 b を処理するときで $B \to b$, $B \to b$, $A \to a$, $B \to b$ の後となる．

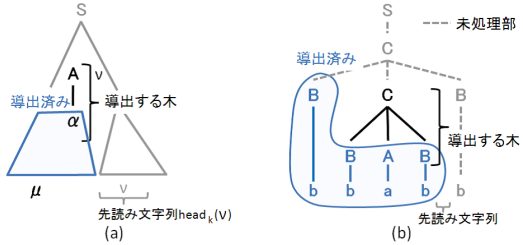

図 3･6　LR(k) 文法の定義と LR(0) 文法の文法規則決定過程

文法例 $G_3=(N_3,T_3,P_3,S_3)=(\{S,A,B,C\},\{a,b\},\{S \to ABB, S \to CB, A \to aA, A \to a, B \to b, C \to aC, C \to a\},S)$ は LR(2) 文法の例である．図 3･7(a), (b) の双方とも始めから 3 文字目の a を処理し，$A \to aC$ または $C \to aC$ のいずれかを先読み文字 *bb* と *b*$（$ は入力の終わりを表す特殊記号）によって決定する．

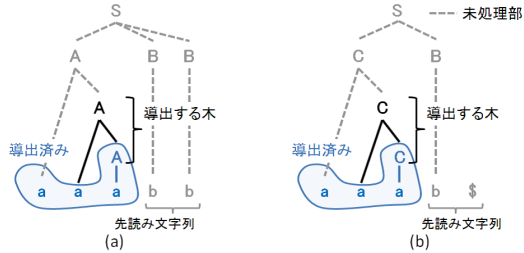

図 3･7　LR(2) 文法の文法規則決定過程

なお，先読み文字列数 k が決めることができるならば，文法の変換アルゴリズム[1]で LR(k) 文法を LR(1) 文法に等価変換可能である．よって，LR(0) 文法と LR(k) 文法，$k \geq 1$，は区別して考えることもある．また，LL(k) 文法は LR(k) 文法の真部分クラスである．

■参考文献

1)　S. Sippu and E. Soisalon-Soininen, “Parsing Theory I,II,” Springer-Verlag, 1990.
2)　井上謙蔵, “コンパイラ プログラム言語処理の基礎,” 丸善, 1994.

■6 群 - 2 編 - 3 章

3-5 文脈自由文法の標準形

（執筆者：藤芳明生）［2008 年 12 月 受領］

本章では，チョムスキー標準形とグライバッハ標準形という文脈自由文法の二つの重要な標準形を紹介する。文脈自由文法が**チョムスキー標準形**であるとは，すべての生成規則が “$A \to BC$” または “$A \to b$” のかたちをしていることである．ここで，A, B, C は非終端記号を表し，b は終端記号を表す．また，**グライバッハ標準形**であるとは，すべての生成規則が “$A \to b\alpha$” のかたちをしていることである．ここで，A は非終端記号，b は終端記号，α は非終端記号の列である．任意の文脈自由文法は，同じ言語を生成するチョムスキー標準形の文脈自由文法に変換できることが知られている．同様に，任意の文脈自由文法を，グライバッハ標準形の文脈自由文法に変換する方法も知られている．

一般論として，ある言語を文法を利用して記述する場合，文法規則のかたちは，なるべく制約されない方が好ましい．しかしながら，文法を処理するアルゴリズムを設計する側にとっては，文法規則のかたちがなるべく簡単であった方が処理しやすい．文脈自由文法に「標準形」と「標準形に変換する方法」が存在することにより，文法利用者は，標準形の制約にとらわれることなく文法が記述でき，文法処理アルゴリズムの設計者は，標準形の文法だけを処理するアルゴリズムの開発に専念することができる．

3-5-1 空規則を含まない文脈自由文法

与えられた文脈自由文法を標準形に変換するためには，元の文法が空規則を含んでいないことが必要である．空規則とは，空語 ε を左辺にもつ，$A \to \varepsilon$ のような生成規則のことである．次の定理より，空規則を含む文脈自由文法は，空規則を含まない文脈自由文法に変換可能であることが分かる．

定理 12 文脈自由文法 G に対し，次の三つの条件を満たす文脈自由文法 G' を構成することができる．(1) G と G' は同じ言語を生成する．(2) G が空語を生成するならば，G' は唯一の空規則 $S \to \varepsilon$ を含む．ただし，S は G' の開始記号であり，S は G' のいかなる生成規則の右辺にも現れない．(3) G が空語を生成しないならば，G' は空規則を含まない．

G' の具体的な構成方法は，形式言語理論の教科書など[1, 2]に載っているので，そちらを参照して頂きたい．注意しなければならない点は，工夫をしないと，構成にかかる時間と得られる文法のサイズが，元の文法のサイズの指数倍になってしまう場合があることである．これを防ぐためには，チョムスキー標準形の文脈自由文法を構成するときに利用されるテクニックを用いて，元の文法に右辺の長さが 3 以上の生成規則があった場合，それを複数の右辺の長さが 2 の生成規則に置き換えておけばよい．すると，構成にかかる時間と得られる文法のサイズは，元の文法のサイズの定数倍で収まる．

文法 G が生成する言語を $L(G)$ とする．$\varepsilon \in L(G)$ の場合，G' から空規則 $S \to \varepsilon$ を取り除くことにより，$L(G) - \{\varepsilon\}$ を生成し，空規則を含まない文法を構成することができる．空語と空規則は特別に扱うこととして，以降，空規則を含まない文脈自由文法だけを考えることとする．

3-5-2 チョムスキー標準形

文脈自由文法が**チョムスキー標準形**であるとは，すべての生成規則が “$A \to BC$” または “$A \to b$” のかたちをしていることである．ここで，A, B, C は非終端記号を表し，b は終端記号を表す．任意の文脈自由文法は，同じ言語を生成するチョムスキー標準形の文脈自由文法に変換できることが知られている．具体的な変換方法は，形式言語理論の教科書など[1, 2]を参照して頂きたい．簡単に流れだけ説明すると，まず，任意の文脈自由文法を空規則を含まない文法に変換する．次に，単位規則（$A \to B$ のようなかたちの生成規則）を除去し，無用な非終端記号を取り除く．最後に，$A \to bCD$ のような規則を，$A \to BCD$ 及び $B \to b$ のような規則に置き換え，$A \to BCD$ のような規則は，$A \to BZ$ 及び $Z \to CD$ のような規則に置き換えることを繰り返す．すると，チョムスキー標準形の条件を満たす文脈自由文法が得られる．上記の流れでチョムスキー標準形の文脈自由文法を構成した場合，元の文法のサイズを n とすると，構成にかかる時間と得られる文法のサイズは，$O(n^2)$ となる．

チョムスキー標準形の文脈自由文法から得られる構文木は，必ず二分木になる．

CYK（Cocke-Younger-Kasami）アルゴリズムなどの構文解析アルゴリズムは，チョムスキー標準形を利用している．

チョムスキー標準形の文脈自由文法から，1 ステップでのスタックの深さの変化がたかだか 1 であるような非決定性プッシュダウンオートマトンを構成できる．

3-5-3 グライバッハ標準形

文脈自由文法が**グライバッハ標準形**であるとは，すべての生成規則が “$A \to b\alpha$” のかたちをしていることである．ここで，b は終端記号，α は非終端記号の列である．任意の文脈自由文法は，同じ言語を生成するグライバッハ標準形の文脈自由文法に変換できることが知られている．グライバッハ標準形の文脈自由文法を構成する場合，まずチョムスキー標準形の文脈自由文法を構成し，そこからグライバッハ標準形を構成する方法がよく利用される．具体的な変換方法は，形式言語理論の教科書など[1, 2]を参照して頂きたい．

グライバッハ標準形は，語彙化可能性という概念と密接な関係がある．形式文法 G が，語彙化（lexicalized）されているとは，G のすべての生成規則の右辺に終端記号が一つ以上現れていることである．文法族 C が語彙化可能（lexicalizable）であるとは，任意の文法 $G \in C$ に対し，語彙化された文法 $G' \in C$ が存在し，$L(G') = L(G)$ となることである．すべての文脈自由文法からなる文法族が語彙化可能であることは，グライバッハ標準形の存在により明らかである．

グライバッハ標準形の文脈自由文法から，空動作を含まない非決定性プッシュダウンオートマトンを構成できる．これは，任意の非決定性プッシュダウンオートマトンに対し，同じ言語を受理する空動作を含まない非決定性プッシュダウンオートマトンが存在することを意味する．

■参考文献

1) J.E. ホップクロフト, J.D. ウルマン 著, 野崎昭弘, 木村泉 訳, “言語理論とオートマトン,” サイエンス社, 1971.
2) J.E. ホップクロフト, R. モトワニ, J.D. ウルマン 著, 野崎昭弘, 高橋正子, 町田元, 山崎秀記 訳, “オートマトン 言語理論 計算論 I,” サイエンス社, 1984.