講　　座

言語理論の最近の話題 II

笠井琢美*・成島　弘**
野口　広***・守屋悦朗****

3. 解析的言語理論

生成文法による言語理論とは別の立場の形式的理論の一つに解析的モデルがある．この理論は Marcus [33]にまとめられているが，**3.1** で最近の二，三の結果を加えて，この大略を解説する [29, 34, 35]．**3.2** では syntactic caluculus などとして知られているカテゴリアル言語につき説明する．カテゴリアル言語は実は自由言語に他ならない．

解析的言語理論においても空でない有限集合 Σ を考え，Σ を**語彙** (vocabulary) と呼び，Σ の元を**語** (word) と呼ぶ．これは解析的言語理論においては言語の構成単位が文字でなく単語であるからであるが，ここでは混乱をさけるため前と同様に Σ を**字母系**と呼び Σ の元を**文字**と呼ぶことにしよう．

生成文法の理論においては形式言語 L を考える場合，Lを生成する文法によって導入されるいろいろな構造が考慮の対象となる．たとえば They are flying planes という文章はそれを生成する文法によって They (are (flying planes)) あるいは They ((are flying) planes) という句構造が与えられる．しかし解析的言語理論においては形式言語L自身から導びかれる構造以外にはなんの構造も仮定しない．むしろLが与えられた時Lを解析することによって種々の構造を明らかにするといった立場にある．

3.1 解析的モデル

解析的モデルにおいては**言語**とは $\Phi=(\Sigma, P, L)$ のことをいう．ただし P は Σ の分割 $P: \Sigma=\bigcup_{i=1}^{n} P_i$, $P_i \cap P_j=\phi (i \neq j)$ で L は Σ 上の形式言語，すなわち Σ^* の部分集合である．各 P_i を P の**セル** (cell) といい Σ の元 a を含むセルを $P(a)$ で示す．したがって Σ の元 a と b に対し $P(a)=P(b)$ であるか $P(a) \cap P(b)=\phi$ である．自然言語における分割 P の通常の解釈は語形変化における単語の分割で，たとえば $P(\text{book})=\{\text{book, books}\}$ となる．

$x=a_1a_2\cdots a_n$ を Σ 上の語とするときセルの有限列 $P(a_1)P(a_2)\cdots P(a_n)$ を語 x の ***P*-構造** (P-structure) といい $P(x)$ で示す．特に $x=\varepsilon$ のとき $P(\varepsilon)=\varepsilon$ と定める．したがって P は Σ^* から 2^{Σ^*} への写像とみることができる．特に x が L の元のとき P-構造 $P(x)$ は **marked** であるという．a と b を Σ の元とするとき任意の P-構造 $\mathcal{P}_1$ と $\mathcal{P}_2$ に対し，もし $\mathcal{P}_1P(a)\mathcal{P}_2$ が marked なら $\mathcal{P}_1P(b)\mathcal{P}_2$ も marked となるとき $P(a) \underset{P}{\rightarrow} P(b)$ あるいは略して $P(a) \rightarrow P(b)$ とかく．$P(a) \rightarrow P(b)$ でありかつ $P(b) \rightarrow P(a)$ であるとき $P(a)$ と $P(b)$ は ***P*-同値** (P-equivalent) であるといい $P(a) \underset{P}{\leftrightarrow} P(b)$ あるいは $P(a) \leftrightarrow P(b)$ とかく．

各セルが Σ のひとつの元からなるような分割を**単位分割** (unit partition) といい E で示す．Σ 上の語 x に対し，x の E 構造 $E(x)$ と語 x とは区別しない．したがって Σ の元 a と b に対し $E(a) \rightarrow E(b)$ とか $E(a) \leftrightarrow E(b)$ とかくかわりにそれぞれ $a \rightarrow b$, $a \leftrightarrow b$ とかく．すなわち $a \leftrightarrow b$ とはすべての Σ^* の元 x と y に対し，$xay \in L$ となる必要十分条件が $xby \in L$ となることである．

さてこれで解析的モデルに関する必要な道具だてをおわる．簡単にいえば解析的モデルでは言語 Φ が与えられたとき関係 $\underset{P}{\rightarrow}, \underset{P}{\leftrightarrow}, \underset{E}{\rightarrow}, \underset{E}{\leftrightarrow}$ に着目して Φ の性質をしらべる．

例　$\Sigma=\{a, b, c, d\}$, $P(a)=P(b)=\{a, b\}$, $P(c)=P(d)=\{c, d\}$, $L=\{ab, ad, ca, cc\}$ とし言語 $\Phi=(\Sigma, P, L)$ を考える．明らかに $b \leftrightarrow d$ であり，a と c に対しては E 同値なものはそれ自身以外にない．Φ の marked な P 構造は $P(a)P(a)$, $P(a)P(c)$, $P(c)P(a)$, $P(c)P(c)$ の四つで，したがって長さ2の P 構造はす

* 京都大学数理解析研究所
** 東海大学理学部数学科
*** 早稲田大学理工学部数学科
**** 電気通信大学電子計算機学科

べて marked となる．よって Σ の任意の元 x と y に対し $P(x)\leftrightarrow P(y)$ となる．一般に Σ の任意の2元 x と y に対し $x\leftrightarrow y$ なら $P(x)\leftrightarrow P(y)$ が成立するような言語 Φ を**適切** (adequate) な言語という．この例の言語は適切な言語である．

導分割 L を Σ 上の形式言語，P を Σ の分割とするとき P の L に関する**導分割** (derivative) P' を

$$P'(a)=\bigcup_{P(b)\leftrightarrow P(a)} P(b),\quad a\in\Sigma$$

とさだめる．今 L を正しい文章からなる（たとえば英語）の集合とし，P を語形変化による分割と解釈すると P' は単語の品詞による分割と考えることができる（[33] 参照）．単位分割 E の導分割を S で示す．すなわち $S(a)=\{b\in\Sigma\mid a\leftrightarrow b\}$ である．さて分割に関する二，三の性質について述べよう．以後 Σ 上の形式言語 L をひとつ固定して考える．

P と Q を Σ の分割とするとき，Σ のすべての元 a に対し $P(a)\subset Q(a)$ となるとき P は Q の**細分**であるといい $P\prec Q$ と書く．$\prec$ は Σ の分割全体からなる集合における半順序となる．P を Q の細分とするとき $P(a)\subset Q(c)$，$P(b)\subset Q(c)$ となる Σ の任意の元 a,b,c に対し $P(a)\leftrightarrow P(b)$ となるなら P は Q の**正則**細分であるという．

定理 1. P と Q を Σ の分割とし，P は Q の細分であるとする．このとき $P'=Q'$ となる必要十分条件は P が Q の正則細分であることである．

補題 P' を P の L に関する導分割とすると $P'(L)=P(L)$ となる．（一般に Q を分割とするとき $Q(L)=\cup_{x\in L}Q(x)$ と定める．）

系 $S(L)=L$.

定理 2. P と Q を Σ の分割とするとき Q が P の導分割となる必要十分条件は Q が $Q(L)=P(L)$ となる分割のうち最大のものとなることである．

系 $Q(L)=L$ となる分割 Q のうち最大のものが単位分割の導分割，すなわち S である．

言語の型 解析的言語理論のおもな研究は言語にいろいろな制限をつけ言語を種々の型に分類し，それらの型のあいだに成立するいろいろな性質について議論することにある．言語の型として重要なものは同質 (homogeneous) な言語と，まえに述べた適切 (adequate) な言語である．

言語 $\Phi=(\Sigma,P,L)$ が同質 (homogeneous) であるとは Σ の任意の元 a と b に対し $S(a)\cap P(b)\neq\phi$ ならば $S(b)\cap P(a)\neq\phi$ となるときをいう．まえの例の言語 Φ は適切ではあるが同質でない言語である．$(S(d)\cap P(a)\neq\phi$ ではあるが $S(a)\cap P(d)=\phi.)$

$\Phi=(\Sigma,P,L)$ を言語とするとき分割 R をつぎのように定義する．a と b を Σ の元とするとき b が $R(a)$ の元である必要十分条件は $a_0=a, a_n=b, a_{i+1}\in S(a_i)\cup P(a_i)$，$0\leq i\leq n$ となる Σ の元の列 $a_0,a_1,\cdots,a_n$ が存在することである．

定理 言語 $\Phi=(\Sigma,P,L)$ が適切である必要十分条件は R が P' の細分となることである．

定理 言語 $\Phi=(\Sigma,P,L)$ が適切ならば R は P' の正則細分となる．

言語の同質性についてはつぎが成立する．

定理 言語 $\Phi=(\Sigma,P,L)$ が同質である必要十分条件は Σ の任意の元 a と b に対し $b\in R(a)$ のときまたそのときにかぎり $P(a)\cap S(b)\neq\phi$ が成立することである．

***PS* 同型** 言語 $\Phi_1=(\Sigma_1,P_1,L_1)$ と $\Phi_2=(\Sigma_2,P_2,L_2)$ が ***PS*-同型** (*PS*-isomorphic) であるとは Σ_1 から Σ_2 の上への1対1の写像 f が存在して $P_2(f(x))=f(P_1(x))$，$S_2(f(x))=f(S_1(x))$ となることである．ただし S_1 と S_2 はそれぞれ Φ_1 と Φ_2 に関する単位分割の導分割である．このような写像 f を ***PS*-同型写像**という．一般に同質性は *PS*-同型写像によって保存されるが適切性は保存されるとはかぎらない．すなわち *PS*-同型である二つの言語で一方は適切であるが他方は適切でないといったものが存在する．さて言語 Φ が**絶対適切** (absolutely adequate) であるとは Φ と *PS*-同型なすべての言語が適切となるときをいう．絶対適切な言語に関してはつぎのことがしられている．

定理 言語 Φ が絶対適切である必要十分条件は Φ が同質であることである．

3.2 カテゴリアル言語
Syntactic caluculus

生成文法理論では文法 G は与えられたものとして，その文法がどのような過程で構成されたかは問題としなかった．ここでは文法がどのように構成されるかという機構を数学的に扱かう．たとえば英語を例にとろう．いま正しい英語の文章すべてに s (sentence) という‘型’をあたえる．また固有名詞には n (name) という‘型’をあたえる．また固有名詞でおきかえられるようなすべての語にも n という型をつける．たとえば John, Napoleon の型は n であり，poor John, fresh milk とか milk, rice にも n という型がつけられる

しかし he とか chair には n という型はあたえられない．(たとえば John likes Napoleon という文章で Napoleon を he あるいは chair でおきかえたものはただしい文章とならない．chair は冠詞を必要とする．) さて

John works　　(1)

という文章の型は s であり，John の型は n である．また John を型 n をもつ任意の語でおきかえてもただしい文章になるから works に型 $n\backslash s$ をあたえる．同様に poor John の型は n，John の型は n であるから poor には型 n/n を与える．

一般に α と β を型とするとき α/β, $\beta\backslash\alpha$ も型と認め，α/β という型は右に β 型の語がきたとき α 型の語となるような語にあたえられ，$\beta\backslash\alpha$ という型は左に β 型の語がきたときに α 型となるような語にあたえられる．このようにして型を計算するとつぎの表 3.1 をうる．

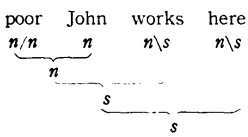
表 3.1

John→n	likes→$(n\backslash s)/n$
works→$n\backslash s$	he→$s/(n\backslash s)$
poor→n/n	him→$(s/n)\backslash s$
here→$s\backslash s$	

(一般に語 x が型 α をもつことを $x\to\alpha$ とかく．)

このように Σ の各元に型をあたえること，すなわち型の辞書をつくることは文法を構成するということにほかならない．

たとえば poor John works here という文章がただしい文章であることはその型を計算するとつぎのように s となることよりわかる．



さて以上のことを厳密に述べる．

定義　C を有限集合とするとき，C 上の**カテゴリー系** $\tilde{C}$ をつぎのように定める．

(a)　$C\subset\tilde{C}$.

(b)　$\alpha,\beta\in\tilde{C}$ なら (α,β), (α/β), $(\alpha\backslash\beta)$ は $\tilde{C}$ の元である．

C の元を**原始カテゴリー**あるいは**原始型** (primitive type) といい，$\tilde{C}$ の元を**カテゴリー**あるいは型 (type) とよぶ．

型 $(\alpha\cdot\beta)$ を $(\alpha\beta)$ とかく．またいちばん外側のかっこは省略する．

さて，いま Σ 上のある語には原始型がつけられているものと仮定する．このときつぎの (i), (ii), (iii) にしたがって語に型をつける．(ここでは空語は考えない．したがって語とは Σ^+ の元を意味する．)

(i)　語 x が型 α をもち語 y が型 β をもつなら語 xy に型 $\alpha\beta$ をあたえる．

(ii)　型 β をもつ任意の語 y に対し語 xy が型 α をもつなら x に型 α/β をあたえる．

(iii)　型 β をもつ任意の語 y に対し語 yx が型 α をもつなら x に型 $\beta\backslash\alpha$ をあたえる．

さて α と β を型とするとき $\alpha\to\beta$ で型 α をもつ任意の語は型 β をもつことを示す．さらに $\alpha\rightleftarrows\beta$ で $\alpha\to\beta$ かつ $\beta\to\alpha$ を示す．するとつぎが成立する．

(a)　$\alpha\to\alpha$.

(b)　$(\alpha\beta)\gamma\to\alpha(\beta\gamma)$,　(b′)　$\alpha(\beta\gamma)\to(\alpha\beta)\gamma$.

(c)　$\dfrac{\alpha\beta\to\gamma}{\alpha\to\gamma/\beta}$,　(c′)　$\dfrac{\alpha\beta\to\gamma}{\beta\to\alpha\backslash\gamma}$.

(d)　$\dfrac{\alpha\to\gamma/\beta}{\alpha\beta\to\gamma}$,　(d′)　$\dfrac{\beta\to\alpha\backslash\gamma}{\alpha\beta\to\gamma}$.

(e)　$\dfrac{\alpha\to\beta\quad\beta\to\gamma}{\alpha\to\gamma}$.

(a) と (b), (b′) が成立することはあきらかであろう．(c) は「もし $\alpha\beta\to\gamma$ なら $\alpha\to\gamma/\beta$ となる」とよむ．(c′) も同様，$\alpha\beta\to\gamma$ を仮定し語 x が α 型をもつとする．すると型 β をもつ任意の語 y に対し xy は型 γ をもつ．したがって x は型 γ/β をもつ．(d′) は (d) と同様であるから (d) を示す．(d) は「$\alpha\to\gamma/\beta$ なら $\alpha\beta\to\gamma$」とよむ．$\alpha\to\gamma/\beta$ を仮定し x と y がそれぞれ型 α,β をもつとしよう．すると xy は型 γ をもち，$\alpha\beta\to\gamma$ となる．(e) は「$\alpha\to\beta$ で $\beta\to\gamma$ なら $\alpha\to\gamma$ となる」とよむ．これは定義よりあきらかである．

さて (a), (b), (b′) を公理とし (c) から (e) を inference rules とするような system を **associative syntactic caluculus** [30] とよぶ．この system はつぎが成立する．

(I)　$(\alpha/\beta)\beta\to\alpha$,　$\beta(\beta\backslash\alpha)\to\alpha$.

(II)　$(\alpha\backslash\beta)/\gamma\rightleftarrows\alpha\backslash(\beta/\gamma)$.

(III)　$(\alpha\backslash\beta)(\beta/\gamma)\to\alpha/\gamma$.
$(\alpha\backslash\beta)(\beta\backslash\gamma)\to\alpha\backslash\gamma$.

(IV)　$\alpha\to\beta/(\alpha\backslash\beta)$,　$\alpha\to(\beta/\alpha)\backslash\alpha$.

(I) は (a) より $\alpha/\beta\to\alpha/\beta$, (d) より $(\alpha/\beta)\beta\to\alpha$ となり成立する．(II) と (III) の証明は少し長くなるので省略する．(IV) は (I) と (c) からえられる．一般にこの system では，有名な Gentzen [28] による手法を

用いてつぎの定理がえられる.

定理 α と β を任意の型とするとき $\alpha\to\beta$ であるかどうかは決定可能である.

もう一度例にもどってつぎの文章を考えよう.

John (likes Jane) (2)

n $(n\backslash s)n$ n

ここで likes Jane は(1)の works と同じ型 $n\backslash s$ であり, したがって likes の型は $(n\backslash s)/n$ となる. (2)のかわりに

(John likes) Jane (3)

を考えると likes には型 $n\backslash(s/n)$ があたえられる. このことが一般に成立すること, すなわち $(n\backslash s)/n \rightleftarrows n\backslash(s/n)$ となることを意味する. また John などの n 型の語は代名詞 he の型 $s/(n\backslash s)$ をもつことが予想されるがこれも(Ⅳ)より $n\to s/(n\backslash s)$ となることからわかる. (Ⅲ)の意味は次節の例で言及する.

$\alpha_1, \alpha_2, \cdots, \alpha_n$ を型とするとき積 $\alpha_1\alpha_2\cdots\alpha_n$ で型 $(\cdots((\alpha_1\alpha_2)\alpha_3)\cdots\alpha_n)$ を示す. しかし(b), (b′)より $\alpha_1, \cdots, \alpha_n$ を任意の順序で構成した型 α に対し

$\alpha \rightleftarrows (\cdots((\alpha_1\alpha_2)\alpha_3)\cdots\alpha_n)$

となることがわかる. これが成立しないような system, すなわち(a)だけを公理とし(c), (c′), (d), (d′), (e) を inference rules とするような system を **non-associative syntactic caluculus** という.

カテゴリアル文法

いま Σ の各元に型があたえられたとする. このとき(Ⅰ), (Ⅱ), (Ⅲ), (Ⅳ) を書きかえ規則だとおもうと Σ 上の語が正しい文であるかどうかが判定できる. たとえば表 3.1 より

he likes him $\Rightarrow (s/(n\backslash s))((n\backslash s)/n)((s/n)\backslash s)$

$\Rightarrow (s/(n\backslash s))(n\backslash(s/n))((s/n)\backslash s)$ (Ⅱより)

$\Rightarrow (s/(n\backslash s))(n\backslash s)$ (Ⅲより)

$\Rightarrow s$ (Ⅰより)

となり, he likes him が正しい文章であることが結論される. このようにして(Ⅰ), (Ⅱ), (Ⅲ), (Ⅳ)を書きかえ規則とするものを自由カテゴリアル文法, (Ⅰ)だけを書きかえ規則とするものをカテゴリアル文法とよぶ. 厳密にいうとつぎのようになる.

定義 **カテゴリアル文法** (categorial grammar) とは $G=(\Sigma, C, s, U, \Rightarrow)$ のことをいう. ただし Σ と C は有限集合で s は C の元, U は Σ の各元 a に対し $\tilde{C}$ の有限部分集合 $U(a)\subset\tilde{C}$ を対応させる関係である. $\Rightarrow$ は(Ⅰ)によって定まる $\tilde{C}$ 上の関係, すなわち

$\Gamma\beta(\beta\backslash\alpha)\Omega \Rightarrow \Gamma\alpha\Omega$

$\Gamma(\alpha/\beta)\beta\Omega \Rightarrow \Gamma\alpha\Omega$

で定まる関係である. また

$L(G)=\{w\in\Sigma^+ \mid \exists\alpha\in U(w))\alpha \overset{*}{\Rightarrow} s\}$

を G によって受理される**カテゴリアル言語**という.

注 $w=a_1\cdots a_n$ とするとき $U(w)$ は $\{\alpha_1\cdots\alpha_n \mid \alpha_i\in U(a_i)\}$ なる型の集合をあらわす. 型 $\alpha(\beta\gamma)$ と $(\alpha\beta)\gamma$ は区別しない. $\overset{*}{\Rightarrow}$ は $\Rightarrow$ の反射推移閉包を示す.

自由カテゴリアル文法 (free categorial grammar) とは $G=(\Sigma, C, s, U, \underset{f}{\Rightarrow})$ のことをいう. ただし Σ, C, s, U はカテゴリアル文法のときと同じ, $\underset{f}{\Rightarrow}$ は(Ⅰ), (Ⅱ), (Ⅲ), (Ⅳ)によって定義される $\tilde{C}$ 上の関係である. この時も $L(G)$ が同様に定義され, $L(G)$ のことを**自由カテゴリアル言語**という.

定理 (Bar-Hillel) カテゴリアル言語の族は自由言語の族と一致する.

定理 (Cohen) 自由カテゴリアル言語の族はカテゴリアル言語の族と一致する.

Grammatical Inference

カテゴリアル言語のおもな目的はあたえられた言語 L に対しそれを生成(受理)するような文法 G の構成であった. しかしこの方法も機械的に実行するとなれば種々の困難さをともなう. これらについては文献をあげるにとどめるが, とくにこの方面の survey としては [24] を参照されたい.

4. 自由文法の拡張

生成文法理論の創始者である N. チョムスキー自身が初期の論文で指摘しているように, 自由文法は自然言語を完全に記述するための道具としては非常に不完全なものである. この不完全さの一部を補うものとして彼は変換文法 (transformational grammar) を提唱した. また, **2.2** で述べたように自由言語は ALGOL 型言語と同等であるとはいうものの, 実際のプログラミング言語 ALGOL の中には自由文法であらわせない文が存在することも知られている. このように, プログラミング言語の記述の道具としても欠陥のある自由文法が何故最も深く研究されてきたかという理由の一つには, 単純でわかり易く, 数学的取り扱いが容易であるということが挙げられる.

生成能力だけを問題にするならば, 自由文法はきわめて弱体である. たとえば, $\{a^nb^nc^n \mid n\geq 1\}$ とか $\{w\cdot w \mid w\in\Sigma^*\}$ とかいうような比較的簡単な形の言語が自由文法では定義できない. もちろん, これらは文脈

文法で定義することができるのではあるが，文脈文法を用いるということは，言語記述の単純明解さ，あるいは扱かい易さを損なうものである．

これを克服しようとする試み，すなわち自由文法の単純さを失なわずにそれを拡張しようとする試みは60年代の後半から現われ始めた．

拡張の主な方法は自由文法の production の適用順序にいろいろな方法で制限をくわえるものである（外部制御型）．この範疇に属する代表的なものは [4], [5], [6] である．これに対し，外部から導出をコントロールするのではなく，文法自身の内にある種の制御機構が備わっている（たとえば，各変数に何らかの記憶機能をもたせる）ような文法を考えることができる（内部制御型）．こういう考え方での拡張の代表的なものが [3] である．

4.1 外部制御型拡張

外部から導出をコントロールする方法として最も簡単なものは，有限オートマトンでプロダクションの適用順序を指定するという方法である．

つぎに述べる行列文法 (matrix grammar) [1] は自由文法の拡張としては最も古いものであり，外見的には有限オートマトンが現われていないが，その一例といえる．

定義 $G=(N,\Sigma,P,S)$ を自由文法とする．G 上の**行列文法** $\bar{G}$ とは対 (G,M) のことをいう．ここで，M はつぎのような行列ルールの有限集合である．

$$[\Pi_1,\Pi_2,\cdots,\Pi_n],\quad n\geq 1,\quad \Pi_i\in P\quad (1\leq i\leq n).$$

$\bar{G}$ はつぎのようにして $L(G)$ の部分集合 $L(\bar{G})$ を生成する．$\bar{G}$ において $w\overset{M}{\Rightarrow}w'$ と書けるのは G において $w\overset{\Pi_1}{\Rightarrow}w_1\overset{\Pi_2}{\Rightarrow}w_2\overset{\Pi_3}{\Rightarrow}\cdots\overset{\Pi_m}{\Rightarrow}w'$ でありかつ $[\Pi_1,\Pi_2,\cdots,\Pi_m]\in M$ となるときのみである．ただし，$w_1\overset{\Pi_2}{\Rightarrow}w_2$ などとかいたのは $\Pi_2\in P$ が $w_1\overset{\Pi_2}{\Rightarrow}w_2$ なる導出に使われたプロダクションであることを示している．$\overset{M^*}{\Longrightarrow}$ を $\overset{M}{\Rightarrow}$ の反射推移閉包として

$$L(\bar{G})=\{w\in\Sigma^*\mid S\overset{M^*}{\Longrightarrow}w\}$$

を行列言語という．

行列文法はかなりよく研究されている．その興味ある部分族の研究については [9][10] が参考になろう．

$L(\bar{G})$ を $\{w\in\Sigma^*\mid S\overset{\alpha}{\Rightarrow}w,\alpha\in M^*\}$ とかくこともできる．すると，次の**制御集合** (control set) **付文法**が容易に導入される．

定義 C を P^* の正則部分集合とする．（べつに正則集合でなくてもよいが，ここでは C は正則集合に限っておく．）制御集合 C をもつ文法 G によって生成される言語は

$$L_c(G)=\{w\in\Sigma^*\mid S\overset{\alpha}{\Rightarrow}w,\alpha\in C\}$$

と定義される．

実は，[13] で導入された制御集合付文法というのは，拡張文法を目指したものではなかったし，また，G における導出も leftmost derivation に話を限っている．

次に述べる状態文法 (state grammar) は日本の研究者によって導入されたものであり [6]，有限オートマトンによる導出のコントロールがよくわかる．

定義 **状態文法**は6組 $G=(K,N,\Sigma,P,p_0,S)$ で示される．N,Σ,S は自由文法と同じ意味をもつ．K は状態の有限集合で，P はつぎのような形のプロダクションの有限集合である．

$$(p,A)\to(q,x),\quad p,q\in K,\ A\in N,\ x\in V^*.$$

$w_1=uAv, w_2=uxv$ で $(p,A)\to(q,x)$ が P の元であるとき $(p,w_1)\Rightarrow(q,w_2)$ とかける．

$$L(G)=\{w\in\Sigma^*\mid(p_1,S)\overset{*}{\Rightarrow}(q,w),q\in K\}$$

を状態言語という．

上の状態文法の導出の定義は [6] のオリジナルなものとやや異なっている．[6] の定義に従がえば，状態言語の族は文脈言語の族と一致する（$x\in V^+$ に制限すると）のであるが，ここでは上の定義に従がったものを状態言語ということにする．このことについては後でも触れよう．最後にプログラム文法について述べよう．

定義 **プログラム文法** (programmed grammar) とは $G=(N,\Sigma,J,P,S)$ のことである．N,Σ,S は自由文法の場合と同じものを意味する．J はプロダクションの番号 (label) の有限集合で，プロダクション（P の元）はつぎのような形をしている．

$$(r)A\to x\quad S(V)F(W).$$

$A\to x$ を核 (core) といい，これは自由文法のプロダクションの形をしている．r はプロダクションの番号であり各プロダクションごとに異なる．V,W は J の部分集合でそれぞれ成功域 (success field)，失敗域 (failure field) とよばれる．

$\Rightarrow$ は次のように定義される．$(r)A\to xS(U)F(W)$ を P の元とする．$(r,w_1)\Rightarrow(s,w_2)$ とかけるのはつぎのいずれかの場合である．

(1) $w_1=uAv$, $w_2=uxv(u\in(V-\{A\})^*)$ で $s\in U$.

(2) $w_1\in(V-\{A\})^*$, $w_2=w_1$ で $s\in W$.

(1)はプロダクション r の適用に成功したのでつぎのプロダクションが U から選ばれ，(2)は失敗したので何もしないでつぎのプロダクションは W から選ばれていることを意味する．

$$L(G)=\{w\in\Sigma^*|(1,S)\overset{*}{\Rightarrow}(t,w),t\in J\}$$

をプログラム言語という．

常に $U=W$ であるようなプログラム文法を**無条件遷移** (unconditional transfer) **プログラム文法** という興味深い文法である．

例 つぎのようなプロダクションをもつプログラム文法は $\{\underline{n}ha^n|n\geq1\}$ を生成する．$\underline{n}$ は整数 n の2進表示である．

(1) $S\to1SB$ $S(\{3\})$ $F(\phi)$
(2) $S\to0S$ $S(\{3\})$ $F(\phi)$
(3) $A\to BB$ $S(\{3\})$ $F(\{4\})$
(4) $B\to A$ $S(\{4\})$ $F(\{1,2,5\})$
(5) $S\to h$ $S(\{6\})$ $F(\phi)$
(6) $A\to a$ $S(\{6\})$ $F(\phi)$

この他に各部制御型文法としては，scattered context 文法 [7]，unordered scattered context 文法 [12]，random context 文法 [8]，periodically time-variant 文法 [5]，ordered 文法 [2]，ストリング文法 [40] などがある．

4.2 内部制御型拡張

indexed 文法 [3] は，直観的には各変数が pushdown store のような記憶機能をもつような自由文法であると考えることができる．プロダクション $A\to u_0B_1u_1\cdots B_nu_n$ $(B_i\in N,u_i\in\Sigma^*)$ は単に A を $u_0Bu_1\cdots B_nu_n$ でかきかえるだけでなく，変数 A の記憶内容に従って各 B_i の記憶内容が指定されるようになっている．この指定の仕方が pushdown store のような形式をとると思えばよい．

定義 indexed 文法 とは $G=(N,\Sigma,F,P,S)$ のことである．N,Σ,S は自由文法の場合と同じ．F は flag の有限集合で，各 flag は $B\to y(B\in N,y\in V^*)$ なる index プロダクションの有限集合である．P は次のようなプロダクションの有限集合である．

$$A\to x;\ A\in N,\ x\in(NF^*\cup\Sigma)^*.$$

関係$\Rightarrow$をつぎのように定義する．$w_1\Rightarrow w_2(w_i\in(NF^*\cup\Sigma)^*)$ とかけるのはつぎのいずれかが成り立つ場合である．

(1) $w_1=\alpha A\theta B$; $\alpha,\beta\in(NF^*\cup\Sigma)^*$, $A\in N$, $\theta\in F^*$
$(A\to X_1\varphi_1X_2\varphi_2\cdots X_m\varphi_m)\in P$, $X_i\in V$, $\varphi_i\in F^*$
$w_2=\alpha X_1\varphi_1X_2\varphi_2\cdots X_m\varphi_m\beta$
ここに，
$$\varphi_i=\begin{cases}\varepsilon & (X_i\in\Sigma\text{ のとき})\\ \psi_i\theta & (X_i\in N\text{ のとき}).\end{cases}$$

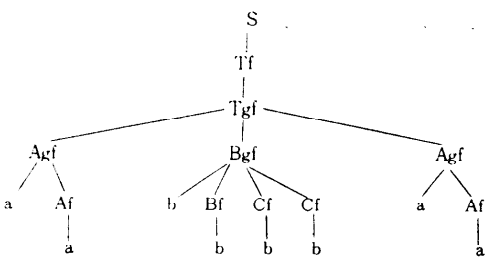
(2) $w_1=\alpha Af\theta\beta$; $\alpha,\beta\in(NF^*\cup\Sigma)^*$, $A\in N$, $f\in F$, $\theta\in F^*$
$(B\to X_1X_2\cdots X_m)\in f$, $X_i\in V$
$w_2=\alpha X_1\varphi_1X_2\varphi_2\cdots X_m\varphi_m\beta$
ここに
$$\varphi_i=\begin{cases}\varepsilon & (X_i\in\Sigma\text{ のとき})\\ \theta & (X_i\in N\text{ のとき})\end{cases}$$

$L(G)=\{w=\Sigma^*|S\overset{*}{\Rightarrow}y\}$ を indexed 言語という．

indexed 文法の大きな利点の一つは，自然なかたちで導出木を定義できることである．たとえば，$G=(\{S,A,B,C,D\},\{a,b\},\{f,g\},P,S)$ なる indexed 文法を考えてみる．

$$P=\{S\to Df,D\to Dg,D\to ABA\},$$
$$f=\{A\to a,B\to b,C\to b\},$$
$$g=\{A\to aA,B\to bBCC,C\to bC\}$$

とすると，$L(G)=\{a^nb^{n^2}a^n|n\geq1\}$ であるが，たとえば，$a^2b^4a^2$ の導出木はつぎのように書ける．



さて，indexed 文法では，各変数の記憶機構が pushdown stack であると考えられたが（記憶内容は各変数の右側に添えられた flag の f,g などである．）この記憶機構を general counter structure なるものに拡張することによって indexed 文法の拡張 (**general indexed 文法**) がえられる [39]．これによって [39] では indexed 言語とその拡張の階層をあたえている．

また **2.2** で自由言語に対する $uvwxy$ 定理を述べたが，[37] はこれを (one way) stack 言語にまで拡張した．さらに，それが日本の研究者によって indexed 言語にまで拡張された [38]．紙面の都合で詳細にふ

れることはできないが，この indexed 文法に対する $uvwxy$ 定理をもちいることによって，$\{a^{n^n} | n \geq 1\}$ などが indexed 言語でないことや indexed 文法に対する finiteness 問題が可解であることなどが証明できる [38].

正則集合 ↔ 有限オートマトン，自由言語 ↔ pushdown オートマトンなどのように，言語とオートマトンの間の密接な関係がしられているが，indexed 言語には nested stack オートマトン [21] が対応する．同様に general indexed 言語には monitored pushdown オートマトンが対応する [39].

自由言語を拡張するのに，自由文法を拡張するのではなく，pushdown オートマトンを拡張することによる方法が考えられる．**スタック** (stack) **言語**はその一例で，スタック・オートマトンによって定義されたものであり [17], [18], indexed 言語の部分族をなす．最近，このスタック言語を文法によって characterize することが行なわれている [19], [20]．たとえば，[19] の**スタック文法**は indexed 文法とよく似ている．詳細は上記論文を参照されたい．

4.3 導出の制限

自由文法においては，任意に書きかえを許す導出方法と，最左端の変数にかぎって書きかえを許す方法 (leftmost derivation(1)) とでは実質的な差異はないが，一般に拡張文法を考えるときにはこの制限は考慮する必要がある．

たとえばプログラム文法は $(r)A \to xS(U)F(W)$ によって x にかきかえられる A はもっとも左側の A でなければならない．このように，いくつも現われる A のうちでは最左側の A にプロダクションを適用する方式を leftmost interpretation(2)，どこの A でも自由にかきかえられるという方式を free interpretation(3) ということにする．

制御集合つき文法においても左側導出に限るか否かは本質的な違いをもたらす．また状態文法でも，オリジナルな導出の定義は，状態 p のもとでかきかえが許される変数は“p のもとでプロダクション適用が可能な変数のうちでもっとも左側のもの”という制限(4)がついている．この制限をつけると文脈言語と一致するが，つけないと行列言語と一致する，という大きな違いが生じる．実際，[6]ではこの制限をゆるめていくことによって（適用可能な変数のうち左側から n 番目までの変数の書きかえを許すことによって言語族 $\mathcal{L}_n$ が定義される）自由言語と文脈言語のあいだに無限の階層を導入している．((1) (2) (3) (4) は違うことに注意.)

さて，このように導出そのものの定義を本質的に変えてみることを考えよう．

同時導出 [42] というのは，プロダクション $A \to x$ を全ての A に一度に適用してしまうような導出である．つまり $(A \to x) \in P$ のとき，$w_1 \Rightarrow w_2$

$w_1 = u_1 A u_2 A \cdots A u_n$, $u_i \in (V - \{A\})^*$. は

$w_2 = u_1 x u_2 x \cdots x u_n$

を意味する．このように $\Rightarrow$ を定義すると自由文法から，$\{a^i b^j a^i b^j | i, j \geq 1\}$ のような自由言語でない言語を生成できる反面，Dyck 言語などは生成できない．

プッシュダウン・スタックが first-in-first-out 方式であるのに対して，first-in-last-out 方式のスタックを Queue スタックという．自由言語がプッシュダウン・スタック・テープをもったオートマトンに対応しているのに対して，Queue スタック・テープをもったオートマトンに対応するような文法を考えることができる．記号処理や情報検索を背景として，**転送文法**[41] はこのアイデアにもとづいて考えられた．単純転送文法の定義する言語の族は自由言語の族と incomparable になる．たとえば $\{ww | w \in \Sigma^*\}$ や $\{a^n b^n c^n | n \geq 1\}$ など単純転送文法によって容易に生成されるが，$\{ww^R | w \in \Sigma^*\}$ は単純転送言語ではない．さらに単純転送言語は Parikh map によって半線形集合となり，文脈言語族に真に含まれることがわかっている．

最後に，導出に関連した研究として導出の長さに制限を加えることによって言語を分類するこころみがある [16], [43] ことも注意しておこう．これによっても自由言語と文脈言語のあいだに無制限の階層が導入される．これらは **2.2** で述べた自由言語の分類などとともに言語の複雑さによる分類研究の範疇に属する．

4.4 言語間の関係

今まで各種の拡張言語などを説明してきたが，これらのあいだの関係についてふれる前に，$\{a^n b^n a^n \ n \geq 1\}$ のような自由言語でない言語がこれらの文法でどのように定義できるかをみてみよう．

たとえば，行列文法ではつぎのような行列ルールをもつものを考えればよい．

(1) $[S \to ABC]$,

(2) $[A \to aA, B \to bB, C \to aC]$,

(3) $[A \to a, B \to b, C \to a]$.

$a^3 b^3 a^3$ はつぎのように生成される．

$S \overset{M}{\Rightarrow} ABC$　　(1)をつかう

$\overset{M}{\Rightarrow} aAbBaC$ (2) 〃

$\overset{M}{\Rightarrow} a^2Ab^2Ba^2C$ (2) 〃

$\overset{M}{\Rightarrow} a^3b^3a^3$ (3) 〃

状態文法では,

(1) $(p_0, S)\to(p_1, ABC)$,

(2) $(p_1, A)\to(p_2, aX)$,

(3) $(p_2, B)\to(p_3, bY)$,

(4) $(p_3, C)\to(p_1, aC)$,

(5) $(p_1, A)\to(p_4, a)$,

(6) $(p_4, B)\to(p_5, b)$,

(7) $(p_5, C)\to(p_1, a)$

とすると

$(p_0, S)\Rightarrow(p_1, ABC)$ (1)をつかう

$\Rightarrow(p_2, aABC)$ (2) 〃

$\Rightarrow(p_3, aAbBC)$ (3) 〃

$\Rightarrow(p_1, aAbBaC)$ (4) 〃

…… (2)(3)(4)をくり返してつかう.

$\Rightarrow(p_1, a^{n-1}Ab^{n-1}BaC)$

$\Rightarrow(p_4, a^nb^{n-1}Ba^{n-1}C)$ (5)をつかう

$\Rightarrow(p_5, a^nb^na^{n-1}C)$ (6) 〃

$\Rightarrow(p_1, a^nb^na^n)$ (7) 〃 .

indexed 文法では,

(1) $S\to aAfa$,

(2) $A\to aAga$,

(3) $A\to B$,

flag: (4) $f=B\to b\}$,

(5) $g=\{B\to bB\}$

とすると

$S\Rightarrow aAfa$ (1)

$\Rightarrow aaAgfaa$ (2)

…… (2)

$\Rightarrow a^nAg^{n-1}fa^n$ (2)

$\Rightarrow a^nBg^{n-1}fa^n$ (3)

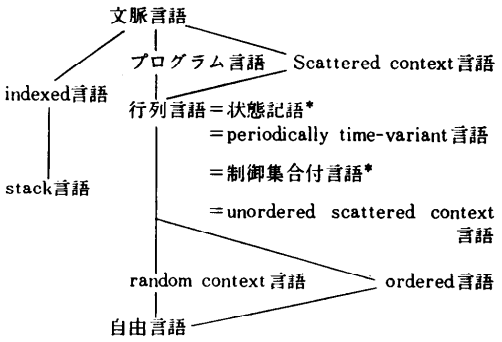
$\Rightarrow a^nbBg^{n-2}fa^n$ (5)

…… (5)

$\Rightarrow a^nb^{n-1}Bfa^n$ (5)

$\Rightarrow a^nb^na^n$ (4).

一見すると行列文法がもっとも単純のようにみえるが, 一概にはそのように断定することはできない. 今まで述べた拡張文法はそれぞれ長所短所があってこれらを比較することはむずかしい. 生成の能力だけを問題にするならつぎのダイヤグラムのようになる.



* は free interpretation

すべて ε-free 自由文法上で定義されているとする.

図 4.1

また無条件遷移プログラム言語 (free interpretation の下で) と行列言語のあいだには, ちょうど Dyck 言語と自由言語の間にあるような関係があることもしられている [11].

参考文献 II

1) S. Abraham: Some questions of phrase structure grammars, Computational Linguistics 4 (1965), 61–70.
2) I. Fis: Grammars with partial ordering of the rules, Information and Control 12(1968), 415–425.
3) A.V. Aho: Indexed grammars—an extension of context-free grammars, J. Assoc. Comput Mach. 15 (1968), 647–671.
4) D. J. Rosenkrantz: Programmed grammars and classes of formal languages, J. Assoc. Comput. Mach. 16 (1969), 107–131.
5) A. Salomaa: Periodically time-variant context-free grammars, Information and Control 17 (1970), 294–311.
6) T. Kasai: An hierarchy between context-free and context-sensitive languages, J. Comput. System Sci. 4 (1970), 492–508.
7) S. Greibach and J. Hopcroft: Scattered context grammars, J. Comput. System Sci. 3 (1969), 233–247.
8) A. P. J. Van der Welt: Random context languages, Symp. on Formal Languages, Oberwolfach, Germany, 1970.
9) R. Siromoney: On equal matrix languages, Information and Control 14 (1969), 135–151.
10) O. H. Ibarra: Simple matrix languages, Information and Control 17 (1970), 359–394.
11) E. Moriya: Some remarks on state grammars

operating under the free interpretation, 1972.
12) O. Mayer: Some restrictive devices for context-free grammars, Information and Control 20 (1972), 69-92.
13) S. Ginsburg and E. H. Spanier: Control sets on grammars, Math. Systems Theory 2 (1968), 166-186.
14) A. Salomaa: On grammars with restricted use of productions, Ann. Acad. Sci. Fenn. Series A 1, 454.
15) B. Brainerd: An analog of a theorem about context-free languages, Information and Control 11 (1968), 561-567.
16) R. V. Book: Time-bounded grammars and their languages, J. Comput. System Sci. 5 (1971), 397-429.
17) S. Ginsburg, S. A. Greibach and M. A. Harrison: Stack automata and compiling, J. Assoc. Comput. Mach. 14 (1967), 172-201.
18) S. Ginsburg, S. A. Greibach and M. A. Harrison: One-way stack automata, J. Assoc. Comput. Mach. 14 (1967), 389-418.
19) R. W. Ehrich and S. S. Yau: Two-way sequential transduction and stack automata, Information and Control 18 (1971), 404-446.
20) M. A. Harrison and M. Schkolnick: A grammatical characterization of one-way nondeterministic languages, J. Assoc. Comput. Mach. 18 (1971), 148-172.
21) A. V. Aho: Nested stack automata, J. Assoc, Comput. Mach. 16 (1969), 383-406.
22) A. W. Bierman: An interactive finite-state language learner, first USA-Japan Computer Conference.
23) A. W. Biermann: "A Gramatical Inference Program for Linear Languages," Fourth Hawaii International Conference on System Sciences, Honolulu, Hawaii, Jan. 12-14, 1971.
24) A. W. Biemann and J. A. Felman: "A Survey of Results in Grammatical Inference," International Conference on Frontiers of Pattern Recognition, University of Hawaii, Honolulu, Hawaii, Jan. 18-20, 1971.
25) J. A. Feldman: "Some Decidability Results on Grammatical Inference and Complexity," Information and Control 20 (1972), 244-262.
26) Y. Bar-Hillel, C. Gaifman and E. Shamir: On categorial and phrase-structure grammars, Bull. Res. Council Israel 9F, 1-16.
27) Y. Bar-Hillel: A quasiarithemetical notation for syntactic description, Language 29 (1953), 47-58.
28) G. Gentzen: Untersuchugen über das logische schlissen, Math. Z. 39 (1934), 176-210, 405-431.
29) T. Kasai: On analytic languages, Bull. Math. de la R. S. de Roumanie Tome 14 (62), nr. 1, (1970).
30) J. Lambek: The mathmatics of sentence structure, Amer. Math. Monthly 65 (1958), 154-169.
31) J. Lambek: Contributions to a mathematical analysis of the English verb-phrase, J. Canad. Ling. Assoc. 5 (1959), 83-89.
32) J. Lambek: On the calculus of systactic types, Proc. Symp. Appl. Math. 12 (1961), 166-178.
33) S. Marcus: Algebraic Linguistics; Analytic Models, Academic Press, New York, 1967.
34) B. Zelinka: Sur les langues absolument bien aéquats, Revue roum. math. pures appl. 12 (1967), 737-739.
35) B. Zelinka: Sur le PS-isomorphisme des Langues, Zeitschrift math. Logik Grundl. Math. 12 (1966), 263-265.
36) J. M. Cohen: The equivalence of two concepts of categorial grammars, Information and Control 10 (1967), 475-484.
37) W. F. Odgen: Intercalation Theorems for Pushdown and store and Stack Languages, (1968) Thesis, Stanford University, also in 1st ACM Symp. Theory on Computing (1969).
38) 林健志: Indexed grammar の木構造について——$uvwxy$ 定理の拡張, 京大数理解析研講究録 156 (1972), 69-82.
39) 西沢輝秦: General indexed grammar と monitored pushdown stack acceptor, 京大数理解析研講究録 123 (1971), 146-169.
40) 伊藤英則, 稲垣康善, 福村晃夫: String controlled grammar, 京大数理解析研講究録 123 (1971), 103-113.
41) 大原茂之, 野島晋, 成島弘: 照合機械, S_0 型転送文法のアクセプター——1 型キューオートマトン——, 転送文法, 電子通信学会論文誌 55 D (1972), 826-827, 56 D (1973), 381-382, 56 D (1973), 440-441.
42) 鳥居宏次, 有沢誠: 同時導出による句構造言語, 電子通信学会論文誌 C (1971), 124-131.
43) 五十嵐善英, 本多波雄: 導出に制限のある Context Sensive 言語, 電子通信学会論文誌 52-C (1969), 595-602.

(昭和 48 年 9 月 27 日受付)