講　　座

言語理論の最近の話題 III

笠井琢美*・成島　弘**
野口　広***・守屋悦朗****

5. 言語の抽象—AFL—

言語理論の発展に伴って，すでに述べたように数多くの形式言語の族がいろいろな形で導入され研究されてきた．この過程を通していろいろな点がクローズアップされてくる．一つは形式言語（とそれに深く関係するオートマトン）の種々の族を統一的に取り扱える一般的な理論を確立したいという要求である．他方で，いろいろな言語の研究を進めていくうちに，それらが共通にもっている基本的な性質が多数見出されてきた．さらにある種の演算は他の演算と独立ではないというようなことも知られていた．これらのことから，ある種の演算は言語にとって基本的なものであり，その基本的な性質を満たすような言語の族を抽象的に考えることにより一般的な理論が展開できるのではないかということが推測される．この背景の下にAFL の概念が生まれた〔1〕.

定義 **AFL** (abstract family of languages) とは次の諸条件を満たす抽象的な言語族 $(\Sigma, \mathcal{L})$ のことである（Σ は省略して $\mathcal{L}$ だけで示すことがある）.

（1） Σ は記号 (symbol) の可算無限集合.

（2） $\mathcal{L}$ の各元 L に対して有限集合 Σ_1 が存在して，$L \subseteq \Sigma_1^*$ となる（$\mathcal{L}$ は言語の族）.

（3） $\mathcal{L}$ は次の6つの演算で閉じている：$\cup$（和集合，$\cdot$（積），$+$ (Kleene 演算)，h^{-1}（逆準同型写像），ε-free な h（準同型写像），$\cap R$（正則集合との共通部分）.〔Iの1参照．以後，これらの演算に対しては，$\cup, \cdot, +, h^{-1}, h, \cap R$ などの略記号を用いる.〕

（4） $\mathcal{L}$ の元で空集合でないものが少なくとも一つ存在する.

$\mathcal{L}$ が ε-free とは限らない任意の準同型写像で閉じている時，これを **full FAL** という．（3）の条件がない時，$\mathcal{L}$ を単に**言語族**という.

AFL は形式言語族の抽象化である．これが十分容認できるものであることは大部分の言語族が (full) AFL〔以下 AFL と full AFL の場合を同時に，full AFL の場合はカッコ内に記して，述べる〕となることに存する．たとえば，正則集合の族，自由言語の族，文脈言語の族，スタック言語の族，indexed 言語の族，句構造言語（O型言語）の族等々．しかし，このような抽象化は言語理論の本来の目的（自然言語論やプログラミング言語の理論に何らかの実際的な貢献をすること）から脇道にそれてしまっているのではないかという批判がないこともない.

5.1 言語上の演算と AGL

まず言語の演算について触れておこう．はじめに (ε-free) 正則集合の族は最小の (AFL) full AFL であることを次の定理に述べる.

定理 full AFL (AFL) は全ての正則集合 (ε-free 正則集合) を含む．(任意の言語族は，$h, h^{-1}, \cap R$ で閉じていれば全ての正則集合を含んでいる.)

定義 **列変換機** (squential transducer) とは $M=(K, \Sigma, \Delta, \lambda, p_0, F)$ のことである．ここに

（1） K, Σ, Δ はそれぞれ状態，入力，出力の有限集合.

（2） λ は $K\times\Sigma^*\times K$ の有限部分集合から Δ^* の有限部分集合族への写像である.

（3） $p_0 \in K$ は初期状態.

（4） $F \subseteq K$ は終局状態（受理状態）の集合.

λ の値域が Δ^+ の有限部分集合であるとき，M は **ε-free** であるという．また，λ が $K\times\Sigma\times K$ から Δ^* の有限部分集合族への写像であるとき，M を特に **gsm** (generalized sequential machine) という．特に，$\#(K)=1$（状態が1つ）でかつ λ が $K\times\Sigma\times K \to \Delta^*$ なる写像である時が**準同型写像** (homomorphism)

* 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

である.

$w\in\Sigma^*$ に対して,

$$M(w)=\{\lambda(p_0,a_1,p_1)\lambda(p_1,a_2,p_2)\cdots\lambda(p_{n-1},a_n,p_n)\mid n\geq 1,\ w=a_1a_2\cdots a_n(a_i\in\Sigma^*),\ p_i\in K,\ p_n\in F\}$$

$$M(L)=\bigcup_{w\in L}M(w)$$

によって, M による変換を定義する. M を 2^{Σ^*} から 2^{Δ^*} への写像をみるとき, その逆写像 $M^{-1}: 2^{\Delta^*}\to 2^{\Sigma^*}$ を次のように定義する. $w\in\Delta^*$, $L\subseteq\Delta^*$ に対して

$$M^{-1}(w)=\{z\in\Sigma^*\mid w\in M(z)\},$$

$$M^{-1}(L)=\{z\in\Sigma^*\mid \text{ある } w\in L \text{ が存在して } z\in M(w)\}.$$

特に M が gsm の時, M^{-1} を**逆 gsm** という. このとき, 必ずしも $M^{-1}(M(L))=M(M^{-1}(L))=L$ は成り立たないので M^{-1} は M の真の意味での逆写像ではないことに注意.

次にいくつかの演算を定義する.

定義 R の正則集合, L, L_1, L_2 を任意の言語とする.

$/R$ (正則集合による右商): $L/R=\{w\mid wy\in R \text{ が存在して } y\in L\}$.

$R\backslash$ (正則集合による左商): $R\backslash L=\{w\mid yw\in R \text{ が存在して } y\in L\}$.

Init: $\mathrm{Init}(L)=\{w\neq\varepsilon\mid wy \text{ が存在して } y\in L\}$.

Sub: $\mathrm{Scb}(L)=\{w\neq\varepsilon\mid uwv \text{ が存在して } u,v\in L\}$.

Fin: $\mathrm{Fin}(L)=\{w\neq\varepsilon\mid wy \text{ が存在して } y\in L\}$.

Shuf (二つの言語の混ぜ合わせ): $\mathrm{Shuf}(L_1,L_2)=\{x_1y_1\cdots x_ny_n\mid x_1\cdots x_n\in L_1, y_1\cdots y_n\cdots L_2\}$.

定義 Σ を字母系とする. Σ の各元 a に対して $\tau(a)$ はある字母系 Σ_a に対して Σ_a^* の部分集合とする. $\tau(\varepsilon)=\{\varepsilon\}$ であり, かつ $x=x_1\cdots x_n(x_i\in\Sigma)$ に対して $\tau(x)=\tau(x_1)\cdots\tau(x_n)$ であるとき, τ を**代入** (substitution) という. 各 $a\in\Sigma$ に対して $\tau(a)$ が正則集合, 有限集合, ε を含まない ($\varepsilon\notin\tau(a)$) とき, これらをそれぞれ**正則代入**, **有限代入**, **ε-free 代入**という.

定理 $\mathcal{L}$ を full AFL とすると, $\mathcal{L}$ は次の演算や写像で閉じている.

(1) 列変換機写像, (2) gsm 写像

(2) 逆 psm 写像, (4) $/R$ および $R\backslash$

(5) Init, Sub および Fin, (6) 正則代入

(7) 正則集合との混ぜ合わせ (Shuf (L_1,L^2) で L_1 または L_2 が正則集合の場合).

((4), (5)を除く他は, 演算を ε-free にすることなどにより, AFL に対しても成り立つ.)

定理 $\mathcal{L}$ を [full] AFL とする.

(1) $\mathcal{L}$ が $\cap$ (共通部分) で閉じているならば, $\mathcal{L}$ は ε-free 代入 [代入] で閉じている〔15〕.

(2) $\mathcal{L}$ が $\cup$ で閉じている必要十分条件は, $\mathcal{L}$ が Shuf で閉じていることである〔2〕.

定理 $\mathcal{L}_1, \mathcal{L}_2$ を full AFL とすると, $\mathcal{L}_1\hat{\partial}\mathcal{L}_2$ も full AFL である ($\hat{\partial}$ の定義は **5.3** 参照)〔10〕.

さて次に AFL の 6 つの演算の独立性および種々の演算の関係について触れておこう〔21〕, 〔31〕.

定義 準同型写像 h が **length-preserving** とは, 任意の w に対して $|h(w)|=|w|$ となることである.

c-代入とは $\tau(a)=c^*ac^*$ なる代入のことである (c は特別な記号).

k を正整数とする. 言語の族 $\mathcal{F}$ が **k-limited erasing** で閉じているとは, $L\subseteq\{\Sigma\{\varepsilon,c,c^2,\cdots,c^k\}\}^*$ ($c\notin\Sigma$ とする. これらの演算については論文 I. 1 参照) なる $\mathcal{F}$ の各元 L に対して $h(L)\in\mathcal{F}$ なることである. ここに h は $h(a)=a(a\neq c)$, $h(c)=\varepsilon$ となる準同型写像とする.

言語の族 $\mathcal{F}$ が **ε-free** とは, $\mathcal{F}$ の各元 L が ε-free. すなわち $\varepsilon\notin L$ なることである.

定理 $\mathcal{F}$を任意の言語族とする.

(1) $\mathcal{F}$ が $\cup, +$, ε-free h, h^{-1}, $\cap R$ で閉じていれば, $\mathcal{F}$ は $\cdot$ でも閉じている.

(2) $\cdot, *$, ε-free $h, h^{-1}, \cap R$ で閉じていれば, $\cup$でも閉じている.

(3) $\mathcal{F}$ が ε-free で, $\cdot$, ε-free h, h^{-1} で閉じていれば $\cap R$ でも閉じている.

(4) h^{-1}, $\cap R$, length-preserving h で閉じていれば ε-free h でも閉じている.

(5) c-代入, k-limited erasing, length-preserving h, $\cup R$ (正則集合との和集合), $\cap R$ で閉じていれば h^{-1} でも閉じている.

(6) 有限代入と $\cap R$ で閉じていれば gsm 写像でも閉じている.

(7) ε-free 代入, k-limited erasing, $\cup$, $\cap R$ で閉じていれば逆 gsm 写像でも閉じている.

(8) 有限代入, $\cap R$ で閉じていれば $/R$ でも閉じている.

定理 (1) $*$は $\cap$, $\cdot$, $\cup R$, h^{-1}, h と独立である. すなわち$*$は後者の 5 つから導かれない.

(2) ε-free h は $\cup$, $\cdot$, $\cap R$, h^{-1}, $*$ と独立である.

(3) h^{-1} は $\cup, \cdot, \cup R, *, h$ (または ε-free h) と独立である.

(4) h は $\cup, \cdot, \cap R, h^{-1}$, ε-free $h, *$ と独立である.

以上をまとめると,

定理 $\mathcal{F}$ を ε-free 言語族とする. $\mathcal{F}$ が AFL である必要十分条件は, $\mathcal{F}$ が $\cdot, +$, ε-free h, h^{-1} で閉じていることである.

定理 次のいずれかが成り立てば $\mathcal{F}$ は AFL である.

(1) $\mathcal{F}$ は $\cup, +, h^{-1}$, ε-free h, $\cap R$ で閉じている.

(2) $\mathcal{F}$ は $\cdot, *, h^{-1}$, ε-free $h, -\{\varepsilon\}, \cup\{\varepsilon\}$ で閉じている.

(2) $\mathcal{F}$ は任意の $a\in\Sigma$ に対して a^* を含み, 代入と $\cap R$ で閉じている.

(4) $\mathcal{F}$ は ε-free で, 任意の $a\in\Sigma$ に対して a^+ を含み, ε-free 代入, $\cup R$, k-limited erasing で閉じている.

最後に, 各種の演算での閉包性を表にしておく.

	正則集合	自由言語	文脈言語	O型言語
$\cap$	Yes	Yes	Yes	Yes
$\cdot$	Yes	Yes	Yes	Yes
$*(+)$	Yes	Yes	Yes(+)	Yes
$\cup$	Yes	No	Yes	Yes
Σ^*-L	Yes	No	?(Σ^+-L)	No
$\cap R$	Yes	Yes	Yes	Yes
代入	Yes	Yes	Yes(ε-free)	Yes
gsm	Yes	Yes	Yes(ε-free)	Yes
$/R$	Yes	Yes	No	Yes

5.2 AFA

前節では言語族の抽象化について述べたが, この節ではオートマトンの族の抽象化 (AFA〔1〕) について考え, それと AFL との関係について述べる. この総合報告ではオートマトン理論については原則として触れないことにしているので, ごく簡単にまとめることにする. なお, ここで述べる AFA と同じようなオートマトン理論の統一化の試みは **baloon オートマトン**による Hopcroft 等の研究〔16〕,〔30〕がある. 〔23〕〔25〕には AFL との関係が述べられている.

定義 **AFA** (abstract family of (one-way nondeterministic) acceptors) とは次を満たす acceptor (オートマトン) の抽象的な族 $(\Omega, \mathcal{D})$ のことである. (Ω を略して $\mathcal{D}$ と略記することがある.)

(1) $\Omega=(K, \Sigma, \Gamma, I, f, g)$, ここに,

(a) K, Σ は可算無限集合 (それぞれ state, input を表わす).

(b) Γ, I は空でない集合 (それぞれ補助記憶テープに使われる記号, 動作命令 (instruction) を表わす).

(c) $f: \Gamma^*\times I\to\Gamma^*\cup\{\phi\}$ は写像 (f は補助記憶テープの内容を変換する関数である).

(d) g は Γ^* から Γ^* の有限部分集合への写像で, $g(\varepsilon)=\{\varepsilon\}$ かつ, $\varepsilon\in g(\gamma)$ となるのは $\gamma=\varepsilon$ の場合のみであるような写像である (g は補助記憶テープからの情報をとり出す関数である).

(e) $g(\Gamma^*)$ の各元 γ に対して ($g(X)$ は $\bigcup_{x\in X} g(x)$ を表わす), I の元 1_γ が存在して, $g(\gamma')\ni\gamma$ となるような任意の γ' に対して $f(\gamma', 1_\gamma)=\gamma'$ を満たす.

(f) I の各元 u に対して次の性質を満たす Γ の有限部分集合 Γ_u が存在する: もし $\Gamma_1\subseteq\Gamma$, $\gamma\in\Gamma_1^*$ かつ $f(\gamma, u)\neq\phi$ なら $f(\gamma, u)\in(\Gamma_1\cup\Gamma_u)^*$, すなわち各 $\gamma\ni\Gamma^*$ に対して $f(\gamma, u)$ の元は γ に属するか Γ_u に属する.

(2) $\mathcal{D}$ は **acceptor** と呼ばれる $D=(K_1, \Sigma_1, \delta, q_0, F)$ の集合である. ここに,

(a) K_1, Σ_1 はそれぞれ K, Σ の有限部分集合で, $F\subseteq K_1$, $q_0\in K_1$ である.

(b) δ は $K_1\times(\Sigma_1\cup\{\varepsilon\})\times g(\Gamma^*)$ から $K_1\times I$ の有限部分集合への写像で

$G_D=\{\gamma\mid$ ある q, a に対して $\delta(q, a, \gamma)\neq\phi\}$

が有限集合となるようなものである.

(1)(e)の 1_γ は identicty instruction であり, その目的は acceptor が状態を変えたり入力を読んだりしても, 補助記憶テープの内容を変えないでおくことにある.

(1)(f)は acceptor D が与えられた時, 使われる補助テープ用の記号が有限集合から選ばれることを保証する.

例 プッシュダウンオートマトンの AFA $(\Omega, \mathcal{D})$ は次のように定まる. $\Omega=(K, \Sigma, \Gamma, I, f, g)$ で, $I=\Gamma^*$, $g(\varepsilon)=\{\varepsilon\}$, $g(\gamma Z)=\{Z\}$, $f(\varepsilon, \gamma)=\gamma$, $f(\gamma Z, u)=\gamma u$ ($Z\in\Gamma, \gamma\in\Gamma^*, u\in\Gamma^*$). g はプッシュダウンテープが空かどうか ($g(\varepsilon)=\{\varepsilon\}$) および, もし空でない時はその一番上 (一番右側) の記号が何であるか ($g(\gamma Z)=\{Z\}$) の情報を与える. f はプッシュダウンテープの内容を変換する. $1_\varepsilon=\varepsilon$, $1_Z=Z$, $\Gamma_\varepsilon=\phi$ であり, もし $u=Z_1\cdots Z_k(Z_i\in\Gamma)$ ならば $\Gamma_u=\{Z_1, \cdots,$

$Z_k\}$ である.

さて, acceptor の動作とそれが受理する集合を記述しよう. $\vdash_D$ (acceptor Dがわかっている時は $\vdash$ と略記する) は $K_1\times\Sigma_1^*\times\Gamma^*$ 上の関係で次のように定義される. $a\in\Sigma_1\cup\{\varepsilon\}$ に対して, $\overline{\gamma}$ と u が存在して

$\overline{\gamma}\in g(\gamma)$, $(p', u)\in\delta(p, a, \overline{\gamma})$, $f(\gamma, u)=\gamma'$

を満たすならば

$(p, aw, \gamma)\vdash_D(p', w, \gamma')$

と書く. 次に $\vdash^t$ を次のように定義する.

(i) 任意の $(p, w, \gamma)\in K_1\times\Sigma_1^*\times\Gamma^*$ に対して $(p, w, \gamma)\vdash^0(p, w, \gamma)$ である.

(ii) (p'', w'', γ'') が存在して, $(p, w, \gamma)\vdash^t(p'', w'', \gamma'')\vdash(p', w', \gamma')$ となる時, $(p, w, \gamma)\vdash^{t+1}(p', w_1, \gamma')$ である.

$\vdash^*$ は $\vdash$ の反射推移閉包, すなわち $(p, w, \gamma)\vdash^*(p', w', \gamma')$ となるのは, ある $k\geq 0$ に対して $(p, w, \gamma)\vdash^k(p'w', \gamma')$ となることである.

定義 $(\mathcal{Q}, \mathcal{D})$ を AFA とし, $D=(K_1, \Sigma_1, \delta, q_0, F)$ を $\mathcal{D}$ の元とする. D によって受理 (accept) される集合 $L(D)$ は

$\{w\in\Sigma_1^*\mid(p_0, w, \varepsilon)\vdash_D^*(p, \varepsilon, \varepsilon),\ p\in F\}$

と定義される. また,

$\mathcal{L}(\mathcal{D})=\{L(D)\mid D\in\mathcal{D}\}$

と定義する. $\mathcal{L}(\mathcal{D})$ はオートマトン族 $\mathcal{D}$ によって受理される言語の族である.

例 $\Gamma=\{Z\}$, $I=\{1\}$, $f(\gamma, 1)=\gamma$ $(\gamma\in\Gamma^*)$, $g(\varepsilon)=\{\varepsilon\}$, $g(Z^n)=\phi$ $(n\geq 1)$ とすると, この AFA は補助記憶テープの内容に依存しない (無視している). δ を $K\times(\Sigma_1\cup\{\varepsilon\})\times\{\varepsilon\}$ から $K_1\times\{1\}$ の部分集合族への写像で, $\delta(q, \varepsilon, \varepsilon)=\{(q, 1)\}$ を満たすものとすれば, これは有限オートマトンを表わす. そして $\mathcal{L}(\mathcal{D})$ は正則集合の族と一致する.

さて, 次の定理は AFA と AFL 関係を与える.

定理 $\mathcal{D}$ が AFA なら $\mathcal{L}(\mathcal{D})$ は full AFL である.

有限オートマトン, プッシュダウン・オートマトン, 1-way nondeterministic スタック・オートマトン等の族はそれぞれ AFA となるので, この定理からただちにそれらのオートマトンが受理する言語の族, すなわち, 正則集合, 自由言語, スタック言語の族が AFL となることがわかる. しかし, linear bounded オートマトンの族は AFA をなさない. 文脈言語の族が AFL であることを述べるために次の準備をする.

定義 $\boldsymbol{N}=\{0, 1, 2, \cdots\}$ とし, T を $\boldsymbol{N}$ から $\boldsymbol{N}$ への関数とする. $D=(K_1, \Sigma_1, \delta_1, p_0, F)$ を $\mathcal{D}$ の元とする. 「任意の $w\in\Sigma_1^*$ に対して, $(p_0, w, \varepsilon)\vdash_D^*(p, \varepsilon, \gamma)$ ならば $|\gamma|\leq T(|w|)$ である」という条件を満たす時 $D\in\mathcal{D}_{T(n)}$ である, として $\mathcal{D}_{T(n)}$ を定義する. 整数 $k\geq 0$ に対して $T_k(n)=T(kn)$ であるとし,

$$\mathcal{L}_{T(n)}(\mathcal{D})=\bigcup_{k\geq 0}\mathcal{L}(\mathcal{D}_{T_k(n)})$$

と定義する.

$\mathcal{D}_{T_k(n)}$ は使用される補助記憶テープの長さが T と k によって制限されているような $\mathcal{D}$ 元から成る. 特に $T(n)$ が $T(n)=n$ なる関数であるとき $\bigcup_{k\geq 1}\mathcal{D}_{T_k(n)}$ は linear bounded オートマトンの族である. したがって $\mathcal{L}_{T(n)}=\mathcal{L}(\mathcal{D}_{T(n)})$ は文脈言語の族である.

定理 $\mathcal{D}$ を AFL とする. T が単調増大関数ならば $\mathcal{L}_{T(n)}(\mathcal{D})$ は AFA である.

定義 $\mathcal{D}$ を AFA, k も負でない整数とする.

$\mathcal{D}_k'=\{D\in\mathcal{D}\mid(p, \varepsilon, \gamma)\vdash_D^i(p', \varepsilon', \gamma')$ ならば $i=k\}$

とし, $\mathcal{L}'(\mathcal{D})=\bigcup_{k\geq 0}\mathcal{L}(\mathcal{D}_k')$ と定義する.

$\mathcal{L}'(\mathcal{D})$ の元は **quasi-realealtime** であるといわれる. 実際, $D\in\mathcal{D}_k'$ ならば D は ε-入力だけで k 回より以上続けて動作することはできない. 特に $\mathcal{D}_0'$ は ε-入力では動作しない (real-time) ような acceptor の集合である.

定理 $\mathcal{D}$ が AFA ならば $\mathcal{L}'(\mathcal{D})$ は AFL である.

この節の最後として, 全ての AFL は或る AFA によって定義されることについて述べておく.

定義 Σ を無限集合, $\mathcal{I}\neq\phi$ を $\mathcal{I}\cap(\Sigma\cap\{\varepsilon\})=\phi$ なるような添字の集合とし, $\mathcal{S}=\{S_i\mid i\in\mathcal{I}\}$ を Σ^* の部分集合から成る或る族で次を満たすものとする.

(a) S_i の中には空でないものが少なくとも1つ存在する.

(b) 各 $S_i\in\mathcal{S}$ に対して, $S_i\subseteq\Sigma_1^*$ なる有限集合 $\Sigma_1\subseteq\Sigma$ が存在する.

このとき, $(\mathcal{Q}, \mathcal{D}(\mathcal{S}))$ ($\mathcal{D}(\mathcal{S})$ と略記する) とは次のような AFA である.

(1) $\mathcal{Q}=(K, \Sigma, \Gamma, I, f, g)$.

(2) $\Gamma=\Sigma$.

(3) $I=\mathcal{I}\cup\Sigma\cup\{\varepsilon\}$.

(4) $\begin{cases} g(\varepsilon)=\{\varepsilon\} \\ g(\gamma Z)=\{Z\} \quad (\gamma\in\Sigma^*, Z\in\Sigma). \end{cases}$

(5) $\begin{cases} f(\gamma, Z)=\gamma Z \quad (\gamma\in\Sigma^*, Z\in\Sigma\cup\{\varepsilon\}) \\ f(\gamma, i)=\varepsilon \quad (i\in\mathcal{I}, \gamma\in S_i). \end{cases}$

(6) K は可算無限集合.

定義 $\mathcal{F}(\mathcal{S})$ は $\mathcal{S}$ を含む最小の AFL を $\hat{\mathcal{F}}(\mathcal{S})$ は $\mathcal{S}$ を含む最小の full AFL を表わすものとする.

定理 (1) $\hat{\mathcal{F}}(\mathcal{S})=\mathcal{L}(\mathcal{D}(\mathcal{S}))$.

(2) $\mathcal{F}(\mathcal{S}\cup\{\varepsilon\})=\mathcal{L}'(\mathcal{D}(\mathcal{S}))$.

(3) $\mathcal{F}$ を AFL とする. $\mathcal{F}=\mathcal{L}(\mathcal{D})$ なる AFA $\mathcal{D}$ が存在する必要十分条件は, $\mathcal{F}$ が full AFL なることがある.

定義 $\hat{\mathcal{M}}(\mathcal{M})$ を (ε-free) 列変換機の等分とする. 言語族 $\mathcal{L}$ に対して,

$$\hat{\mathcal{M}}(\mathcal{L})=\{M(L)\mid L\in\mathcal{L}, M\in\hat{\mathcal{M}}\}$$

と定義する. $\mathcal{M}(\mathcal{L})$ も同様に定義される.

5.3 に述べる代入演算子 $\hat{\sigma}, \sigma$ を用いると, $\hat{\mathcal{F}}(\mathcal{S})$, $\mathcal{F}(\mathcal{S})$ は次のように表わすこともできる. ここに, $\mathcal{R}(\mathcal{R}_0)$ は (ε-free) 正則集合の族である.

$$\hat{\mathcal{F}}(\mathcal{S})=\mathcal{R}\hat{\sigma}\hat{\mathcal{M}}(\mathcal{S}),$$

$$\mathcal{F}(\mathcal{S})=\mathcal{R}_0\hat{\sigma}\mathcal{M}(\mathcal{S})=\mathcal{R}\sigma\mathcal{M}(\mathcal{S}).$$

言語理論で対象となる問題の一つに言語の変換がある. 数学的に言えば, 変換機 (transducer) とは acceptor に出力の機能がついたものである. したがって, 変換機の抽象族 **AFT** (abstract family of transducers) $\mathcal{D}^0$ は AFA $\mathcal{D}$ を用いて定義される. AFT については, ここでは詳しく述べない ([1], [22] などを参照) が, たとえば次のようなことが成り立つ.

$\mathcal{F}$ を AFL, $\mathcal{D}$ を AFA とする. $\mathcal{L}(\mathcal{D})$ が $\mathcal{F}$ の元との∩で閉じている必要十分条件は, 任意の $M\in\mathcal{D}^0$ と任意の $L\in\mathcal{F}$ に対して $M(L)\in\mathcal{L}(\mathcal{D})$ となることである.

$\mathcal{F}$ を正則集合の族にとってみれば, その意味はわかるであろう.

なお, 前後するが, AFA に関する最近の研究については[13]等を参照されたい.

5.3 いろいろな AFL

すでに述べた結果 (たとえば full AFL $\mathcal{F}$ に対して, AFA $\mathcal{D}$ が存在して $\mathcal{F}=\mathcal{L}(\mathcal{D})$ となる) からわかるように, AFL は **1-way nondeterministic** オートマトンのクラス (AFA) と深い関係がある. そこで当然次のような問題が生じる. **2-way** オートマトン, あるいは **deterministic** オートマトンのクラスと対応するような AFL と類似のものが考えられないか? 実際そのようなものも研究されている. 次に述べるのは deterministic オートマトンの族と対応するものである[24]. なお, ここでは触れないが, これに関連する研究として[23]等があるので参照されたい.

定義 $L_1\subseteq\Sigma_1^*$, $L_2\subseteq\Sigma_2^*$, $c\in\Sigma-(\Sigma_1\cup\Sigma_2)$ とするとき, L_1cL_2, $cL_1\cup cL_2$, $(L_1c)^*$, $(L_1)\dagger$ をそれぞれ **marked product, marked union, marked** $*$, **marked** $\dagger$ という (marked $*$[$\dagger$] の定義は原論文とは異なる).

定義 \$ を特別な記号とする. **marked gsm** とは $M=(K, \Sigma, \Delta, \lambda, p_0)$ のことである. ここに, K, Σ, Δ, は列変換機のそれと同じものを表わす. また, λ は $K\times(\Sigma\cup\{\$\})$ から $K\times(\Delta^*\cup\Delta^*\$)$ への写像で, 任意の $p\in K$, $a\in\Sigma$ に対して

$$\lambda(p, a)\in K\times\Delta^*, \quad \lambda(p, \$)\in K\times\Delta^*\$$$

を満たすものである. 列変換機の場合と同じように, $w\in L_1\subseteq\Sigma^*$, $L_2\subseteq\Delta^*$ に対して $M(w)$, $M(L_1)$, $M^{-1}(L_2)$ が定義される. ただし, ここでは受理状態 F を必要としていない. ($F=K$ と考えればよい.)

定義 **AFDL** (abstract family of deterministic languages) とは, marked union, marked $*$, および逆 marked gsm (すなわち M^{-1}) で閉じているような言語族 $(\Sigma, \mathcal{L})$ のことである. (AFL の定義を参照. 注意: この 4 つの演算は独立である.) Σ は略することがある.

AFDL という名前は, 後で述べるように, それが 1-way nondeterministic オートマトンが受理する言語の族と一致することに由来する. 正則集合, 自由言語, deterministic なプッシュダウン・オートマトンで受理される言語, 帰納的集合などの族は全て AFDL となる. 文脈言語は marked $*$ で閉じていないので AFDL ではないが, しかし, $\{L, L\cup\{\varepsilon\}\mid L$ は文脈言語$\}$ は AFDL となる.

AFDL の基本的な性質は次のようである: 任意の AFDL は正則集合の族を含んでいる. また, 逆 gsm および正則集合との共通部分をとること ($\cap R$) で閉じている.

さて, AFDL と deterministic オートマトンとの関係を述べよう.

定義 \$ の特別な記号とする. **AFDA** (abstract family of (1-way) deterministic acceptors) とは $(\Omega, \mathcal{D})$ のことである. ここに,

(1) $\Omega(K, \Sigma, \Gamma, I, f, g)$ で,

(a) K, Σ, Γ, I, f は, AFA のそれと同じ.

(すなわち, AFA の定義における(1)(a)～(1)(c)および(1)(f)を満たす.)

(b) $\$\in\Sigma$.

(c) g は $\Gamma^*\to\Gamma^*\cup\{\phi\}$ なる写像で, $g(\gamma)=\varepsilon \Leftrightarrow \gamma=\varepsilon$ という条件を満たす.

(d) 任意の $\gamma\in g(\Gamma^*)$ に対し, $1_\gamma\in I$ が存在して, $\gamma=g(\gamma')$ なる全ての γ' に対して $f(\gamma', 1_\gamma)=\gamma$ を満たす.

(2) $\mathcal{D}$ は acceptor $D=(K_1, \Sigma_1, \delta, q_0, F)$ の集合であり,

(a) K_1, Σ_1, q_0, F は AFA の場合と同じ,

(b) δ は $K_1\times(\Sigma_1\cup\{\varepsilon\}\cup\{\$\})\times g(\Gamma^*)\to(K_1\times I)\cup\{\phi\}$ なる写像で,

$\{\gamma \mid$ ある p, a に対して $\delta(p, a, \gamma)\neq\phi\}$

が有限集合となるようなものである.

(c) 任意の $p\in K_1$, $\gamma\in\Gamma^*$ に対して, (i)ある $a\in(\Sigma_1\cup\{\$\})$ に対して $\delta(p, a, \gamma)\neq\phi$ なら $\delta(p, \varepsilon, \gamma)=\phi$ であり, (ii)$\delta(p, \varepsilon, \gamma)\neq\phi$ なら, 任意の $a\in(\Sigma_1\cup\{\$\})$ に対して $\delta(p, a, \gamma)=\phi$ である.

$D\in\mathcal{D}$ に対して,

$L(D)=\{w\in\Sigma_1^* \mid (p_0, w\$, \varepsilon)\vdash_D^*(p, \varepsilon, \varepsilon),\ p\in F\}$

が定義され, $\mathcal{L}(\mathcal{D})$ も AFA の場合と同様に定義する. また, $\mathcal{L}'(\mathcal{D})$, $\mathcal{L}_{T(n)}(\mathcal{D})$ も同様に定義すると, AFL—AFA の場合と同じような関係が成り立つ.

定理 (1) $\mathcal{D}$ を AFDA とすると, $\mathcal{L}(\mathcal{D})$, $\mathcal{L}'(\mathcal{D})$, $\mathcal{L}_{T(n)}(\mathcal{D})$ は AFDL である. ただし, $T(n)$ は単調増大関数とする.

(2) 言語族 $\mathcal{L}$ が AFDL である必要十分条件は $\mathcal{L}=\mathcal{L}(\mathcal{D})$ なる AFDA $\mathcal{D}$ が存在することである.

言語理論において, 具体的な言語の族を与えるには, (1)文法によるか, または (2) acceptor (オートマトン) によって記述することが多い. ところで, 言語理論にしばしば現われるもう一つの言語の定義あるいは特徴づけの方法は, 「ある言語の族 $\mathcal{S}$ を含み…の性質を満たす最小の言語族」という方法である. たとえば, 正則集合の族は, 「全ての有限集合を含み, $*$, ・, $\cup$で閉じた最小の族」である. この概念は, すでに $\mathcal{F}(\mathcal{S})$ ($\mathcal{S}$ を含む最小の AFL) という形で AFL 理論の中に取り入れられている. このような場合, $\mathcal{S}$ は $\mathcal{F}(\mathcal{S})$ の生成元 (generator) の集合とみることができる.

ところで, Chomsky-Shutzenberger の定理によって自由言語 L は Dyck 言語 D, 正則集合 R と準同型写像 h によって, $L=h(D\cap R)$ と表わされる. h も $\cap R$ も AFL 演算であるから, このことは自由言語の族は一つの生成元 ($\mathcal{S}=\{D\}$) で $\hat{\mathcal{F}}(\mathcal{S})$ と表わせるのではないかということを暗示している. そこで, 次の単項 AFL の概念に到る[2].

定義 $\mathcal{L}$ を AFL とする. $\mathcal{L}$ が**単項** (principal) **AFL** であるとは, ある言語 L が存在して, $\mathcal{L}=\mathcal{L}(\{L\})$ となることである. $\mathcal{F}(\{L\})$ は $\mathcal{F}(L)$ と略記される. $\mathcal{L}$ が単項 **full** AFL (full principal) であるとは, $\mathcal{L}=\hat{\mathcal{F}}(L)$ となることである. Lを**生成元** (generator) という.

単項 AFL という概念は決して異質のものではない. 正則集合, 自由言語, スタック言語, indexed 言語, 文脈言語, O型言語など多くの言語族が単項 AFL となることが知られている[2][9][27].

例 (1) 正則集合の族は明らかに単項 AFL である. その生成は, ϕ, $\{\varepsilon\}$ でない任意の有限集合でよい.

(2) 自由言語の族は Dyck 言語を生成とする単項 AFL である.

(3) 帰納集合 (recursive set) の族は単項 AFL でない.

(4) $\Sigma=\{a_1, \cdots, a_i, \cdots\}$ とし, h を $h(a_i)=a_1{}^i a_2$ なる Σ^* 上の準同型写像とする. $L_1, \cdots, L_n, \cdots$ をO型言語の枚挙とし, L_n のゲーデル数を $g(n)$ とする. $L=\{a_1{}^{g(n)}a_2h(w) \mid w\in L_n(n\geq 1)\}\cup\{\varepsilon\}$ はO型言語であり, O型言語の族はこの L を生成として単項 AFL となる.

単項 AFL $\mathcal{F}(L)$ は次のように表わすことができる. $L\subseteq\Sigma^*$ で $c\notin\Sigma$ とすると,

(1) $\mathcal{F}(L\cup\{\varepsilon\})=\mathcal{M}((Lc)^*)$.

(2) $\hat{\mathcal{F}}(L)=\hat{\mathcal{M}}((Lc)^*)$.

(3) $\mathcal{F}(L\cup\{\varepsilon\})=\{h_2[h_1{}^{-1}((Lc)^*)\cap R] \mid R\in\mathcal{R}$, h_1, h_2 は準同型写像で, h_2 は R 上で ε-limited$\}$.

(4) $\hat{\mathcal{F}}(L)=\{h_2[h_1{}^{-1}((Lc)^*)\cap R] \mid R\in\mathcal{R}$, h_1, h_2 は準同型写像$\}$.

次に AFL の理論によくあらわれる演算を定義する.

定義 $\mathcal{L}, \mathcal{L}_1, \mathcal{L}_2$ を任意の言語族とする時,

$\mathcal{L}_1\wedge\mathcal{L}_2=\{L_1\cap L_2 \mid L_i\in L_i\}$,

$\hat{H}(\mathcal{L})=\{h(L) \mid L\in\mathcal{L}, h$ は準同型写像$\}$,

$\mathcal{L}_1\hat{\partial}\mathcal{L}_1=\{\tau(L_1) \mid L_1\subseteq\Sigma_1^*$ で $L_1\in\mathcal{L}_2$,

τ は $a\in\Sigma_1$ に対して $\tau(a)\in\mathcal{L}_2$ となるような代入$\}$.

と定義する. 上の $\hat{H}(\mathcal{L})$, $\mathcal{L}_1\hat{\partial}\mathcal{L}_2$ において, h, τ を

ε-free としたとき，これを $H(\mathcal{L})$, $\mathcal{L}_1\sigma\mathcal{L}_2$ で表わす．

たとえばつぎのようなことが成り立つ．

定理 $\mathcal{L}_1, \mathcal{L}_2$ が単項 [full] AFL であるとすると

(1) $H(\mathcal{L}_1\wedge\mathcal{L}_2)[\hat{H}(\mathcal{L}_2\wedge\mathcal{L}_1)]$ も単項 (full) AFL である．

(2) $\mathcal{L}_1\sigma\mathcal{L}_2$ [$\mathcal{L}_1\hat{\sigma}\mathcal{L}_2$] も単項 [full] AFL である．

定理 $\mathcal{L}_{r.e.}$ をO型言語の族とするとき，次のことが成り立つ〔6〕〔13〕〔32〕．

(1) $\mathcal{L}$ を 1-counter machine が受理する言語の族とすると，$\mathcal{L}_{r.e.}=\hat{\mathcal{F}}(\mathcal{L}\wedge\mathcal{L})$ である．

(2) $\hat{\mathcal{F}}\cap(L)=\mathcal{L}_{r.e.}$ である．ここに $\hat{\mathcal{F}}\cap(L)$ は L を含み，$\cap$ で閉じた最小の full AFL を表わし，$L=\{a^nb^n|n\geq 0\}$ である．

単項 AFL については〔2〕〔3〕〔6〕などを参照されたい．

さて，ψ を Parikh 写像とするとき，任意の自由言語 L に対して $\psi(L)$ は半線形集合であることはすでに述べた (Parikh の定理)．このようにParikh 写像の下で半線形集合となるような言語を **slip 言語** (language with the semilinear property) という〔18〕．(ある Parikh 写像 ψ_1 で $\psi_1(L)$ が半線形集合となれば，任意の Parikh 写像 ψ_2 で $\psi_2(L)$ は半線形集合となることに注意)．

定義 言語族 $\mathcal{L}$ が **slip 言語族 (slip AFL)** であるとは，($\mathcal{L}$ が AFL であり，かつ) $\mathcal{L}$ の任意の元が slip 言語であることである．

slip AFL の例は，正則集合，自由言語，equal matrix 言語，simple matrix 言語の族などである．

定理 (1) $\mathcal{L}_1, \mathcal{L}_2$ が slip 言語族 (slip AFL) なら，$\mathcal{L}_1\hat{\sigma}\mathcal{L}_2$ も slip 言語族 (slip AFL) である．

(2) $\mathcal{L}$ が slip AFL である必要十分条件は，$\hat{\mathcal{F}}(L)$ が slip AFL となることである．

(3) $\mathcal{L}$ を $\cap R$ および ε-free 有限代入で閉じた slip 言語族とすると，$\hat{\mathcal{F}}(\mathcal{L})=\mathcal{R}\hat{\sigma}(\mathcal{R}_0\hat{\sigma}\mathcal{L})$ である．

(4) $\mathcal{L}$ を slip AFL とする．$\mathcal{L}$ の代入による閉包，すなわち，$\mathcal{L}$ を含み代入で閉じた最小の AFL は slip AFL である．

次に，有界言語 (**2.2** 参照) を生成元とするような AFL について触れておく〔11〕〔19〕．

定義 言語族 $\mathcal{L}$ が**有界 (full) AFL** ((full) bounded AFL) であるとは，有界言語の族 $\mathcal{S}$ が存在して，$\mathcal{L}=\mathcal{F}(\mathcal{S})$ ($\mathcal{L}=\mathcal{F}(\mathcal{S})$) となることである．

定理 (1) $\mathcal{L}$ を有界 AFL とすると，$\hat{\mathcal{F}}(\mathcal{L})$ も有界 AFL である．

(2) $\mathcal{L}_1, \mathcal{L}_2$ を正則集合でない言語を少なくとも一つは含む AFL (これを非正則 AFL という) とすると，$\mathcal{L}_1\sigma\mathcal{L}_2$ および $\mathcal{L}_1\wedge\mathcal{L}_2$ はいかなる有界 full AFL にも含まれていない．

(3) $\mathcal{L}_1$ をある有界 AFL に含まれている非正則 AFL とし，$\mathcal{L}_2$ を任意の AFL とすると，$\mathcal{L}_1\sigma\mathcal{L}_2\subseteq\mathcal{L}_1$ または $\mathcal{L}_2\sigma\mathcal{L}_1\subseteq\mathcal{L}_1$ が成り立つ．

(4) $\mathcal{L}$ がある有界 full AFL に含まれる非正則 AFL ならば，$\mathcal{L}$ は ε-free 代入，$\cap$ のいずれでも閉じていない．

最後に，それ自体は AFL ではないが，AFL に準じる言語族を挙げておく．このうち，semi-AFL はいろいろな形でよく研究されている方である．

定義 **semi-AFL** とは，$\cup$, ε-free h, h^{-1}, $\cup$ で閉じた言語族のことである．任意の h で閉じているとき，これを full semi-AFL という．

定義 **pre-AFL** とは，marked product, marked $\dagger$, h^{-1}, $\cap R$, $\cup\{\varepsilon\}$ で閉じた言語族のことである．

定理 $\mathcal{L}$ を pre-AFL とすると

(1) $H(\mathcal{L})$ 〔$\hat{H}(\mathcal{L})$〕 は 〔full〕 AFL である．

(2) $\mathcal{L}$ が$\cap$で閉じているならば，$\hat{H}(\mathcal{L})$〔$H(\mathcal{L})$〕は$\cap$と〔ε-free〕代入で閉じた AFL である．

以上，AFL 理論について述べてきたが，定義と定理の羅列に終わってしまったようである．AFL 理論の真の理解のためには，定義，定理を最小限に止めて，例を挙げながら背景などについて述べるべきだったように思われる．

詳しい文献表を付したのでそれらを参照されたい．

6. 文法の証明法

CS 文法のような複雑な文法については，それが意図した言語を生成するかどうかを実際に証明してみせることはたびたびやっかいな問題となる．しかし **4.1** で述べたプログラム文法，状態文法などは流れ図の形を書くことができこの流れ図をもとにして最近 Floyd が提唱した方法〔33〕により実際に証明してみせることができる〔34〕．

たとえば **4.4** の状態文法はつぎの図のような nondeterministic な流れ図となる．

さてこの流れ図が任意の $n\geq 1$ に対し $a^nb^nc^n$ を生成することは **4.4** でみたように生成過程をつくってみればわかるが，$a^nb^nc^n$ 以外のものを生成しないことはつぎのようにして示す．

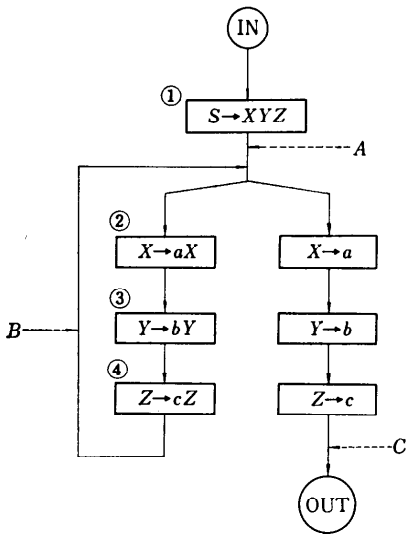

図 6.1

この流れ図の矢図 A, B, C のところに

A: $\{XYZ\}$

B: $\{a^nXb^nYc^nZ \mid n \geqq 1\}$

C: $\{a^nb^nc^n \mid n \geqq 1\}$

といった集合（条件）をつける．さて最初のステップ①で生成される語は A の元である．今②に入る直前に生成される語は $A \cup B$ の元，すなわち $a^nXb^nYc^nZ$，$n \geqq 0$ のかたちであると仮定すると，②，③，④を適用したあとこの語は $a^{n+1}Xb^{n+1}Yc^{n+1}Z$，$n \geqq 0$ となり B の元となる．同様に⑤に入る語は $B \cup A$ の元であるとすると，この語は⑤，⑥，⑦を通過後 C の元となる．したがってこの流れ図で生成される語はすべて C の元となる．

一般に各ステップで入口と出口に条件をつけ，入口の条件をみたす語は出口の条件をみたすことを各ステップについて示せば生成される語はすべて出口の条件をみたすことが示されたことになる．

この例では文法が単純であったため，たいして証明の効果はないが，もっと複雑な言語においては，この技法が有効となるであろう〔34〕．上の言語 $L=\{a^nb^nc^n \mid n \geqq 1\}$ を生成する CS 文法 G に関して $L(G)=L$ となることの厳密な証明が〔Hopcroft & Ullman〕にあるから比較されたい．

さらに高級な言語を生成するために図 6.2 のような分岐条件をつけることができる（プログラム文法）．これは変数 X が直前に生成された語に含まれるとき＋の方向に，含まれないときは－の方向に進むことを

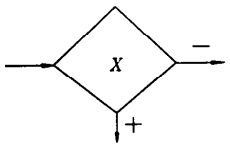

図 6.2

意味する．また各プロダクションを最左に適用するすなわち $X \to u$ なるプロダクションはもっとも左側にあらわれる X に適用する）といった条件をつけたり，さらにいろいろな分岐条件を書くことを許して流れ図の能力を増すこともできる．

謝辞

著者らは早稲田大学数学科の夜久竹夫氏の御協力に対して感謝します．

参考文献 3

1) S. Ginsburg and S. Greibach, Abstract families of languages, *Memoirs Amer. Math. Soc.* 87 (1969), 1-32.
2) S. Ginsburg and S. Greibach, Principal AFL, *J. Comput. System Sci.* 4 (1970), 308-338.
3) J. S. Ullian, Three theorems concerning principal AFLs, *J. Comput. System Sci.* 5(1971) 304-314.
4) S. Ginsburg and J. Goldstein, On the largest full sub-AFL of an AFL, *Math. Systems Theory* 6 (1972), 241-242.
5) R. V. Book, S. A. Greibach and B. Wegbreit, Time-and tape-bounded Turing acceptors and AFL, *J. Comput. System Sci.* 4 (1970), 606-621.
6) S. Ginsburg and J. Goldstein, Intersection-closed full AFL and the recursively enumerable languages, to appear in *Inform. Control.*
7) S. Ginsburg and M. Harrison, On the closure of AFL under reversal, *Inform. Control* 17 (1970), 395-409.
8) S. Ginsburg and J. Hopcroft, Images of AFL under certain families of homomorphisms, *Math. Systems Theory* 5 (1971), 216-227.
9) S. Ginsburg and G. F. Rose, On the existence of generators for certain AFL, *Inform. Sci*, 2 (1970), 431-446.
10) S. Ginsburg and E. H. Spanier, Substitutions in families of languages, *Inform. Sci.* 2 (1970), 83-110.
11) J. Goldstein, Substitution and bounded languages, *J. Comput. System Sic.* 6 (1972), 9-29.
12) S. Greibach, Chains of full AFLs, *Math.*

Systems Theory 4 (1970), 231-242.
13) S. Greibach and S. Ginsburg, Mutitape AFA, *J. Assoc. Comput. Mach.* 19(1972), 193-221.
14) D. L. Lewis, Closure of families of languages under substitution operators, *Proc. Second Ann. ACM Sympo. on Theory of Computing* (1970), 100-108.
15) S. Ginsburg, S. A. Greibach and J. Hopcroft, Pre-AFL, *Memoirs Amer. Math. Soc.* 87 (1969), 41-51.
16) J. Hopcroft and J. Ullman, An approach to a unified theory of automata, *Bell System Techn. J.* 46 (1967), 1793-1829.
17) S. A. Greibach, Syntactic operations on full semi-AFLs, *J. Comput. System Sci.* 6 (1972)
18) S. Ginsburg and E. H. Spanier, AFLs with semilinear property, *J. Comput. System Sci.* 5 (1971), 365-396.
19) J. Goldstein, Abstract families of languages generated by bounded languages, SDC document TM-738/059/00 (1970).
20) S. Greibach, Full AFLs and nested iterated substitution, *Inform. Control* 16(1970), 7-35.
21) S. Greibach and J. Hopcroft, Indepence of AFL operations, *Memoirs Amer. Math. Soc*, 87 (1969), 33-40.
22) O. H. Ibarra, Charactarizations of transductions defined by abstract families of transducers, *Math. Systems Theory* 5 (1971), 271-281.
23) A. V. Aho and J. D. Ullman, A characterization of two-way deterministic classes of languages, *IEEE Conf. Record of Tenth Ann, Symp. on Switching and Automata Theory* (1969), 231-239.
24) W. J. Chandler, Abstract families of deterministic languages, *Proc. ACM Symp. on Theory of Computing* (1969), 21-30.
25) S. Ginsburg and J. E. Hopcroft, Two-way balloon automata and AFL, *J. Assoc. Comput. Mach.* 17 (1970), 3-13.
26) R. Book, S. Ginsburg, O. Ibarra and B. Wegbeit, Tapebounded Turing acceetors and principal AFLs, *J. Comput. System Sci.* 4(1970) 622-625.
27) B. Wegbreit, A generator of context-sensitive longuages, *J. Comput. System Sci.* 3 (1969), 456-462.
28) S. Greibach, A note on undecidable properties of formal languages, *Math. Systems Theory* 2 (1968), 1-6.
29) R. V. Book and B. Wegbreit, A note on AFLs and bounded erasing, *Inform. Control* 19 (1971), 18-29.
30) J. E. Hopcroft and J. D. Ullman, Decidable and undecidable questions about automata, *J. Assoc. Comput. Mach.* 15 (1968), 317-324.
31) J. E. Hopcroft and J. D. Ullman, "Formal Languages and Their Relation to Automata," Addison-Wesley, Reading, Mass. (1969).
32) J. Hartmanis and J. Hopcroft, What makes some language theory problems undecidable, *J. Comput. System Sci.* 4 (1970), 368-376.
33) R.W. Floyd, Assigning meaning to programs, *Symp. Appl. Math.* 19 (1967), 18-32.
34) J. Van Leeuwen, Rule-Labeled Prgrams, Ph. D. dissertation.

(昭和48年10月15日受付)