数理解析研究所講究録
第213巻 1974年 166-179

Bottom-up Generation of Languages

電気通信大学 西澤輝泰

§0. 今例えば, context free grammar $G=(V,\Sigma,S,P)$; $V=\{S,A\}\cup\Sigma$, $\Sigma=\{a,b\}$, $P=\{S\to aSAb,\ S\to b,\ A\to aS,\ A\to a\}$ で生成される言語 L を考える。

L は,

(1) V^* 上の binary relation $\Longrightarrow$ を,

$$\varphi\Longrightarrow\psi \overset{\text{def.}}{\Longleftrightarrow} [(\exists v\in V-\Sigma)(\exists\varphi_0,\varphi_1,\varphi_2\in V^*)\ \varphi=\varphi_1 v\varphi_2 \wedge \psi=\varphi_1\varphi_0\varphi_2 \wedge (v\to\varphi_0)\in P]$$

で定め, relation $\Longrightarrow$ の reflexive, transitive closure を $\overset{*}{\Longrightarrow}$ とするとき,

$$L=\{x\in\Sigma^* \mid S\overset{*}{\Longrightarrow}x\}$$

で定められるが, また, 周知のように, 次のようにも定められる。

(2) 次の条件をみたす, Σ^* の最小の部分集合 L:

$$(\exists L_A\subseteq\Sigma^*)[b\in L\wedge(x\in L\wedge y\in L_A\supset axyb\in L)\wedge a\in L_A\wedge(x\in L\supset ax\in L_A)]$$

(1) と (2) の定義にそれぞれ応じて，L の元の生成過程は，例えば次のようになるであろう。

(1)′ $S \Rightarrow aSAb \Rightarrow aaSAbAb \Rightarrow aabAbAb \Rightarrow aababSbAb \Rightarrow aababbAb \Rightarrow aababbab$

(2)′ $b \in L \longrightarrow \left.\begin{matrix} ab \in L_A \\ b \in L \end{matrix}\right\} \longrightarrow \left.\begin{matrix} abab b \in L \\ a \in L_A \end{matrix}\right\} \longrightarrow aababbab \in L$

ここで，仮りに，(1)′ のような生成過程を top-down の生成，(2)′ のような生成過程を bottom-up の生成と呼ぶことにしよう。

今まで様々な，context free grammar の拡張が考えられているが，その多くは上記にいうところの top-down の生成過程を，いろいろな制御機構をつけ加えて拡張しようとしたものである。ここでは，上記でいう bottom-up の生成過程の拡張を考えてみよ。

さて，上記 (2) のような記述を一般的な形で把えれば，

次のような形式となるであろう。

(*1) 次の条件をみたす最小の集合 L_0 :

$$(\exists L_1)(\exists L_2)\cdots(\exists L_n)\left(\bigwedge_{i=0}^{n}\varphi_i \wedge \bigwedge_{i=0}^{n}\bigwedge_{j=1}^{r_i}\psi_i^{(j)}\right)$$

ただし, $\varphi_i \equiv \bigwedge_{j=1}^{s_i}(x_i^{(j)} \in L_i)$

$$\psi_i^{(j)} \equiv \left(\bigwedge_{k=1}^{t_{ij}} z_k \in L_{l(i,j,k)}\right) \supset f_{ij}(z_1,\cdots,z_{k_{ij}}) \in L_i$$

$(\ 0 \leqq l(i,j,k) \leqq n\ ,\ x_i^{(j)} \in \Sigma^*\ )$

ところで, 次の proposition が成り立つ。

Prop. 1 関数 $f : (\Sigma^*)^k \longrightarrow \Sigma^*$ から自然に導かれる関数 $\tilde{f} : (\mathcal{P}(\Sigma^*))^k \longrightarrow \mathcal{P}(\Sigma^*)$; $\tilde{f}(L_1,\cdots,L_k) = \{ f(x_1,\cdots,x_k) \mid x_i \in L_i \text{ for } 1 \leqq i \leqq k \}$ は連続, 即ち

(i) $L_i \subseteq L'_i$ $(1 \leqq i \leqq k)$ ならば,

$\tilde{f}(L_1,\cdots,L_k) \subseteq \tilde{f}(L'_1,\cdots,L'_k)$

(ii) $L_i^{(0)} \subseteq L_i^{(1)} \subseteq \cdots \subseteq L_i^{(n)} \subseteq L_i^{(n+1)} \subseteq \cdots$ なる集合の列の k 組 $(i = 1, 2, \cdots, k)$ に対し,

$$\bigcup_{n=0}^{\infty} \tilde{f}(L_1^{(n)},\cdots,L_k^{(n)}) = \tilde{f}\left(\bigcup_{n=0}^{\infty} L_1^{(n)},\cdots,\bigcup_{n=0}^{\infty} L_k^{(n)}\right)$$

[証明略]

従って, (*1) は次の表現と等価である。(ただし $\tilde{f}$ の ~ は

省略する。）

（＊2） 次の方程式系をみたす最小の L_0 ：

$$\begin{cases} L_0 = \bigcup_{j=1}^{r_0} f_{0j}(L_{\ell(0,j,1)}, \cdots, L_{\ell(0,j,k_{0j})}) \cup \bigcup_{j=1}^{\Delta_0} \{x_0^{(j)}\} \\ \quad\vdots \\ L_n = \bigcup_{j=1}^{r_n} f_{nj}(L_{\ell(n,j,1)}, \cdots, L_{\ell(n,j,k_{nj})}) \cup \bigcup_{j=1}^{\Delta_n} \{x_n^{(j)}\} \end{cases}$$

この表現を，超言語変数 $\alpha_0, \alpha_1, \cdots, \alpha_n$ を用いて，ALGOL風に次のような形に記述するのも見やすいが，この場合は，第一式の左辺の変数に対応する言語の定義として見ることにする。

（＊3）
$$\begin{cases} \alpha_0 = \sum_{j=1}^{r_0} f_{0j}(\alpha_{\ell(0,j,1)}, \cdots, \alpha_{\ell(0,j,k_{0j})}) + \sum_{j=1}^{\Delta_0} x_0^{(j)} \\ \quad\vdots \\ \alpha_n = \sum_{j=1}^{r_n} f_{nj}(\alpha_{\ell(n,j,1)}, \cdots, \alpha_{\ell(n,j,k_{nj})}) + \sum_{j=1}^{\Delta_n} x_n^{(j)} \end{cases}$$

（ただし，$\Delta_i = 0$ のときは，$+\sum_{j=1}^{\Delta_i} x_i^{(j)}$ の部分は $+\phi$ とかくかまたは何もかかない。）

関数の記述の便宜上，次の記法は断りなしに用いることにする。

1° x と y の concatenation は xy で表す。

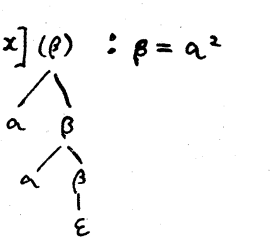
2° x の reverse は x^R で表す。

3° homomorphism $h: \Sigma^* \to \Sigma^*$ で，$a_{i_1}, \cdots, a_{i_m} \in \Sigma$ に対し $h(a_{i_j}) = u_j$ $(j=1, \cdots, m)$; $a \neq a_{i_1}, \cdots, a_{i_m}$ なる $a \in \Sigma$ に対し $h(a) = a$ なる h を $H^{a_{i_1}, \cdots, a_{i_m}}_{u_1, \cdots, u_m}$ で表す。

4° $f: (\Sigma^*)^m \to \Sigma^*$ に対し，$F: (\mathcal{P}(\Sigma^*))^\ell \to \mathcal{P}(\Sigma^*)$; $\ell \leqq m$; $F(L_1, \cdots, L_\ell) = \{ f(x_{i_1}, \cdots, x_{i_m}) \mid x_i \in L_i \text{ for } 1 \leqq i \leqq \ell \}$ (ただし $1 \leqq i_1, \cdots, i_m \leqq \ell$) なる F を，$\lambda x_\ell \lambda x_{\ell-1} \cdots \lambda x_1 [f(x_{i_1}, \cdots, x_{i_m})]$ で表す。

例.1 $\{ a^n b^n a^n \mid n \geqq 0 \}$ の生成。

$$\begin{cases} \alpha = \lambda x [x H^a_b(x) x](\beta) \\ \beta = a\beta + \varepsilon \end{cases}$$



§1. さて，上記 (*3) の形式による言語の記述で，用いる関数の複雑さと，生成される言語の複雑さとの対応関係を考えてみるために，今仮りに，用いる関数が関数族 $\mathcal{F}$ の中から任意に選ばれるとき，得られる言語のなす族を「$\mathcal{F}$ - 言語族」，「$\mathcal{F}$ - 言語族」に属する言語を「$\mathcal{F}$ - 言語」と呼ぶことにする。次の各 propositions は明らかである。

Prop. 2　$\mathcal{F}$ を関数族とするとき，$\mathcal{F}$ の元と定数関数をすべて含み，代入に関して閉じているような最小の関数族を $\tilde{\mathcal{F}}$ とすれば，$\mathcal{F}$-言語族 $=$ $\tilde{\mathcal{F}}$-言語族．

ただし，ここおよび以下で，関数間の「代入」とは次のような操作をいう。即ち，

$f:(\Sigma^*)^{n+1}\longrightarrow\Sigma^*$，$g:(\Sigma^*)^m\longrightarrow\Sigma^*$ に対し，

$h(x_0,\cdots,x_{i-1},x_{i+1},\cdots,x_n,y_1,\cdots,y_m)=f(x_0,\cdots,x_{i-1},g(y_1,\cdots,y_m),x_{i+1},\cdots,x_n)$ なる関数 $h:(\Sigma^*)^{n+m}\to\Sigma^*$ を，f への g の「代入」によって得られる関数という。

Prop. 3　{concatenation}-言語 $\Longleftrightarrow$ context free language

Prop. 4　{primitive recursive function}-言語
$\Longleftrightarrow$ recursively enumerable set

recursively enumerable sets すべての族については，実はもっと単純な関数族で足りる。

Th. 1　{concatenation, gsm-mapping}-言語
$\Longleftrightarrow$ recursively enumerable set

[証明]　L を recursively enumerable set $\subset \Sigma^*$ とする。L が関数としては gsm-mapping と concatenation のみを用いて，§0. (*3) のような形式で記述できることを示す。$L=\phi$ なる自明であるから，$L\neq\phi$ とする。すると，Σ の特定の元 a_0 に対し，$L=\{f(a_0^n)\mid n\geq 1\}$ となる (primitive) recursive function f が存在する。Turing machine T (Σ': set of tape symbols (空白記号も含む)，Q: set of state symbols，q_0: initial state，q_h: halting state) を，この関数 f を計算する machine であって，$q_0a_0^n$ で計算を開始して，$q_hf(a_0^n)$ で計算を停止するような machine とする。$\#$ を，Σ' に含まれない記号として，次のような gsm-mapping を g_1 とする。($g_1:(\Sigma'\cup Q\cup\{\#\})^*\longrightarrow(\Sigma'\cup Q\cup\{\#\})^*$.)

$g_1(xqay\#)=xq'by\#$　if quadruple $aqq'b\in T$

$g_1(xq\#)=xq'b\#$　if 〃　$bqq'b\in T$

$g_1(xqay\#)=xaq'y\#$　if 〃　$aqq'R\in T$

$g_1(xq\#)=xbq'\#$　if 〃　$bqq'R\in T$

$g_1(xbqay\#)=xq'bay\#$　if 〃　$aqq'L\in T$

$g_1(qax\#)=q'bax\#$　if 〃　$aqq'L\in T$

$g_1(xqay\#)=xqay\#$　if $aqq'\alpha\notin T$ for all $q'\alpha$

$g_1(xq\#)=xq\#$　if $bqq'\alpha\notin T$　〃

(ただし上記で, $x, y \in \Sigma'^*$, $a, b \in \Sigma'$.)

また, 次のような gsm mapping を g_2 とする.

$g_2(x\#) = x$

$g_2(ax) = g_2(x)$ $(a \in \Sigma')$

$g_2(xb) = g_2(x)$

$g_2(g_h x) = x$

$g_2(gx) = f(a_0)$ $(g \in Q - \{g_h\})$

(ただし上記で $x \in (\Sigma' \cup Q)^*$)

すると L は次のような系で記述できる.

$$\begin{cases} \alpha = g_2(\beta) \\ \beta = g_1(\beta) + g_0\gamma \\ \gamma = a_0\gamma + a_0\# \end{cases}$$

Cor. 1 任意の recursively enumerable set L に対し, {concatenation, csm-mapping} 一言語 L' と homomorphism h が存在して, $L = h(L')$ となる.

Cor. 2 {concatenation, csm-mapping, homomorphism} 一言語 $\Longleftrightarrow$ recursively enumerable set

Cor. 3 context free language $\rightleftharpoons$ {concatenation,

{com-mapping}-言語 $\rightleftharpoons$ context sensitive language

[証明] 第2の proper implication は 次に述べる定理2による。第1の proper implication は，cfl の族が homomorphism で閉じていることによる。

次に context sensitive languages の族との比較について，

Th. 2 $\mathcal{F}'$ を，語の長さを減じない1変数関数であって linear bounded automaton で計算できるような関数のみからなる任意の関数族とする。$\mathcal{F}'' = \mathcal{F}' \cup \{\text{concatenation}\}$ の，代入 と λ-notation による合成に関する closure を $\mathcal{F}$ とすると，

$\mathcal{F}$-言語 $\rightleftharpoons$ context sensitive language.

[証明] 与えられた $\mathcal{F}$-言語に対し，これを accept する linear bounded automaton をつくることができる。(構成は省略.) 逆が成立しないことの証明には，S. Arikawa の length-growing function の考え方を用いる。§0, (*2) の方程式系の最小解を，

$$\begin{cases} L_i^{(0)} = \phi \\ L_i^{(m+1)} = \bigcup_{j=1}^{r_i} f_{ij}\left(L_{\ell(i,j,1)}^{(m)}, \cdots, L_{\ell(i,j,k_{ij})}^{(m)}\right) \cup \bigcup_{j=1}^{\delta_i} \{x_i^{(j)}\} \end{cases}$$

により，$L_i^{(m)}$ $(i=0,1,2,\cdots\cdots)$ を構成していって，$L_0=\bigcup_{m=0}^{\infty} L_0^{(m)}$ として与えるとき，$\bigcup_{i=0}^{n} L_i^{(m)}$ の中の最長の元の1つを y_m とすれば $(m\geq 1)$，y_{m+1} の長さは y_m の長さの定数倍＋定数でおさえられるから，y_m の長さは ある定数 k_1 と k_2 により，$k_1^m k_2$ でおさえられる。従って，$L_0^{(m)}$ の中の最長の元を z_m とすれば，z_m の長さも $k_1^m k_2$ でおさえられる。従って，L_0 の元を長さの順に並べて $u_1, u_2, \cdots\cdots$ とすれば，z_m の長さは u_m の長さ以上であるから，u_m の長さも $k_1^m k_2$ でおさえられる。長さが k^m より急なのび方（例えば m^m）とする context sensitive language はいくらでもつくれるから，$\mathfrak{F}$-言語でない context sensitive languages が存在する。

Cor. 4 {concatenation, reverse, ε-free gsm-mapping} による後の，$\cup$ と λ-notation による合成 の closure をとっても，この closure を $\mathfrak{F}$ として，

$\mathfrak{F}$-言語族 $\subsetneq$ context sensitive languages の族

§2. 以上で生成能力のだいたいの見当がついたわけであるが，今度はもう少し能力の弱いところで考えて見る。

Prop. 5　{concatenation, homomorphism} — 言語族 $\supsetneq$ context free languages の族

[証明]　$\Sigma = \{a\}$ 上の言語で，$\alpha = h(\alpha) + a$; h は $h(a) = aa$ なる homomorphism ; で規定される言語は $\{a^{2^n} \mid n \geq 0\}$ であって，context free language でない。

しかし長さを増さない homomorphism については事情が異る。実際，次の定理が成り立つ。

Th. 3　$\mathfrak{F}_1 = \{$concatenation, reverse, length-non-increasing homomorphism$\}$ とするとき，

$\mathfrak{F}_1$ — 言語 $\Longleftrightarrow$ context free language

[証明]　$L \subseteq \Sigma^*$ を $\mathfrak{F}_1$ — 言語とし，これを記述する系を　① $[\alpha_0 = \beta_0, \cdots\cdots, \alpha_n = \beta_n]$　とし，系①で用いられる関数は $\mathfrak{F}_1$ の元のみであるとする。length-non-increasing homomorphism は有限個しかないことと，reverse と homomorphism が可換であることに着目する。すべての length-non-increasing homomorphism h に

対し，$[h, \alpha_i]$，$[h^R, \alpha_i]$ なる形の元をすべて新たに超変数にとる。系①の各式 $\alpha_i = \beta_i$ に対応して，すべての length-non-increasing homomorphism h に対し，

$[h, \alpha_i] = h(\beta_i)$，$[h^R, \alpha_i] = h^R(\beta_i)$ なる式をすべて集める。ただしここで $h(\beta_i)$, $h^R(\beta_i)$ は，

1° $h(\pi_1 + \pi_2) = h(\pi_1) + h(\pi_2)$，

$h^R(\pi_1 + \pi_2) = h^R(\pi_1) + h^R(\pi_2)$

2° $h(\pi_1 \cdot \pi_2) = h(\pi_1) \cdot h(\pi_2)$

$h^R(\pi_1 \cdot \pi_2) = h^R(\pi_2) \cdot h^R(\pi_1)$

3° $a \in \Sigma$ に対し，$h^R(a) = h(a)$ （Σ^* の元）

4° $h(h'(\alpha_i)) = [h \circ h', \alpha_i]$，$h(\alpha_i) = [h, \alpha_i]$，

$h(\alpha_i^R) = [h^R, \alpha_i]$，$h^R(h'(\alpha_i)) = [(h \circ h')^R, \alpha_i]$，

$h^R(\alpha_i) = [h^R, \alpha_i]$，$h^R(\alpha_i^R) = [h, \alpha_i]$

により β_i を変換していって，Σ^* の元と超変数と定数のみを concatenation と＋でつないだ式に帰着させたものを表す。これらをすべて集めたものを系②とする。明らかに L は系②により，超変数 $[I, \alpha_0]$（I は identity homomorphism）に対応する言語として定められる。

ところで，λ-notation による合成をみとめると事情はまた異る。簡単のため関数変数の，代入と λ-notation

による合成に関する closure を $\widetilde{\mathcal{F}}$ とするとき，$\widetilde{\mathcal{F}}$ - 言語（族）を complex $\mathcal{F}$ - language (family) と呼ぶことにする。実は，

Prop. 6 complex {concatenation} - language family は，S. Arikawa の Simple Formal System で規定される言語の族に一致する。

従って，

Cor. 5 complex {concatenation, length-preserving homomorphism} - language であって complex {concatenation} - language でない言語が存在する。

[証明] 言語 $\{a^n b^n a^n \mid n \geq 1\}$ は Simple Formal System で規定できない。(S. Arikawa.) しかしこの言語は 5 ページの例 1 でみるように complex {concat., length-preserving homom.} - language である。

Cor. 6 complex {concatenation, reverse} - language であって，complex {concatenation} - language でない言語

が存在する。

[証明] $\{a^n b^{2n} a^n \mid n \geq 1\}$ は $[\alpha = \lambda x[xx^R](\beta), \beta = a\beta b + ab]$ により生成されるから complex {concat., reverse} - language であるが，しかし Simple Formal System で規定し得ない。(証明は $\{a^n b^n a^n \mid n \geq 1\}$ の場合と全く並行)

謝辞. Cor. 6 は，シムポジウム終了後，私信により京大・林健志氏に指摘していただいたものである。ここに記して謝す。

reference

S. Arikawa. Elementary Formal Systems and Formal Languages, Research Report No. 4, 1969, Res. Inst. Fund. Inf. Sci., Kyushu Univ.. (九大理学部紀要にも再録)