言語の認知科学配布資料
形式言語

浅川伸一

2009 年 12 月 1 日

1　文法の形式的概念

チョムスキーは句構造文法の考え方を提唱し，文法理論の考えを押し広めた。ここで文法の形式的な概念を整理しておく。

文法には，名詞，動詞句，文などの統語範疇があり，そこから単語の列が導きだされた。これに対応して，統語範疇を非終端記号 nonterminal symbol または変数 variable と呼び，個々の単語を終端記号 terminal symbol と呼ぶことにする。

次の，変数や記号の間の関係を生成規則 production と呼ぶことにする。名詞句 NP → 形容詞 Adj 名詞 N などのように。ある特別な非終端記号があって，その言語に属すると考えられるすべての終端記号列は，その非終端記号から生成される。この特別な非終端記号のことを文，または開始記号と呼ぶ。

形式的には文法 G を (V_N, V_T, P, S) ここで V_N は非終端記号（変数）の集合，V_T は終端記号の集合，P は生成規則の集合，S は開始記号である。V_N, V_T, P はどれも有限集合である。非終端記号 V_N と終端記号 V_T とをあわせて $V_N \cup V_T = V$ とおく。

空文 empty sentence ϵ とは 0 個の記号列をさす。アルファベット V 上のすべての文の集合を V^* であらわす。また集合 $V^* - \{\epsilon\}$ を V^+ で表す。

生成規則の集合 P は $\alpha \to \beta$ という形の式の集まりである。ここで α は V^+ の中の列であり，β は V^* の中の列である。

$\alpha \to \beta$ が生成規則 P に属しており，γ と δ とが全ての文 V^* の中の任意の列である場合に $\gamma\alpha\delta \underset{G}{\Longrightarrow} \gamma\beta\delta$ である。このとき $\gamma\alpha\delta$ は文法 G において直接に $\gamma\beta\delta$ を導くという。また生成規則 $\alpha \to \beta$ が列 $\gamma\alpha\delta$ に適用され，列 $\gamma\beta\delta$ が得られたともいう。このように $\underset{G}{\Longrightarrow}$ は 2 つの列を第 2 の列が第 1 の列から一個の生成規則の適用によって得られる場合をさす。$\alpha_1, \alpha_2, \cdots, \alpha_{m-1} \underset{G}{\Longrightarrow} \alpha_m$ であるとする。このとき，$\alpha_1 \overset{*}{\underset{G}{\Longrightarrow}} \alpha_m$ と書く。すなわち 2 つの列 α, β が $\alpha \overset{*}{\underset{G}{\Longrightarrow}} \beta$ であるとは，α に生成規則 P を何回か適用して β が得られることをいう。

文法 G によって生成される言語を

$$\{w|w \text{ は } V_T^* \text{ に属し，かつ } S \overset{*}{\underset{G}{\Longrightarrow}} w\} \tag{1}$$

として定義し，$L(G)$ であらわす。いいかえれば，ある列が $L(G)$ に属するのは次の条件が満たされた場合である。

1. その列は，終端記号のみからなる
2. その列は，S から導き得る

より一般的には，文法は以下の 4 つの要素によって定義される。

1. 非終端記号の集合：V_N
2. 終端記号の集合：V_T
3. 生成規則の集合：P
4. 初期記号：σ $(\sigma \in V_N)$

文法はこれら 4 つの要素の組によって

$$G = (V_N, V_T, P, \sigma) \tag{2}$$

と表される。文法 G によって初期記号 σ から生成される文の集合を $L(G)$ と書き，これを文法 G によって生成される言語であるという。すなわち言語 $L(G)$ は

$$L(G) = \{w|w \in V_T^*, \sigma \overset{*}{\Rightarrow} w\} \tag{3}$$

である。これは w は V_T の要素からなる文字列で，σ から導出されるものすべてであることを表している。V_T^* は集合 V_T の任意の長さの記号列であって空列 λ を含む。空列 λ を含まない場合は V_T^+ と表記する。

2　文法の型 チョムスキー階層

書き換え規則の一般形は，

$$\alpha \to \beta,\ (\alpha \in V^+, \beta \in V^*, V = V_N \cup V_T) \tag{4}$$

と書くことができる。ただし，ここで，α には非終端記号の集合 V_N 中の記号がすくなくとも一つ含まれていなければならない。上式 4 は記号列 α が文字列中に存在すれば，これを記号列 β に置き換えることを意味する。そして，α は空列を含まない，その言語に属するすべての文字列の要素であり，β は空列を含んだ全ての文字列からなる集合である。

2.1 0 型文法

初期記号 σ から，この形の生成規則を順次適用していって生成される言語 $L(G)$ を **0 型言語** という。この場合の文法を 0 型文法と呼ぶ。

2.2 1 型文法

式 4 に

$$|\alpha| \leq \|\beta| \tag{5}$$

という制限，すなわち，初期記号の要素の方が書き換え規則を適用した後の要素よりも少ない，という制限をつけた文法を **1 型文法**, あるいは**文脈規定定型句構造文法**, **文脈依存文法**と呼ぶ。

ある文法 $G = (V_N, V_T, P, S)$ で，P のすべての生成規則 $\alpha \to \beta$ が条件 $|\alpha| \leq |\beta|$ をみたすとき，この文法 G を 1 型文法または文脈依存 context sensitive 文法という。文脈依存文法では，ある $\alpha_1, \alpha_2, \beta \in V^*, \beta \neq \epsilon$ および $A \in V_N$ によって $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ のようにあらわされることを要求する。このように制限しても言語の範囲は変わらないことが証明できる。これが文脈依存文法の由来である。

2.3 2 型文法

さらに生成規則，式 4 が以下のように

$$A \to B, \ (A \in V_N, \beta \in V^+) \tag{6}$$

すなわち A から β への書き換え規則のうち A は非終端記号の集合 V_N の要素であり，β は，その言語で生成されるすべての文の集合 (より正確にはそこから空列を除いた集合) V^+ の要素であるという制約を加えた文法のことを **2 型文法**, あるいは**文脈自由型句構造文法**と呼ぶ。

ある文法 $G = (V_N, V_T, P, S)$ の生成規則 $\alpha \to \beta$ について，

1. α は一個の変数であり，
2. β は 空文 ϵ 以外の任意の列である

が満たされている場合，この文法は 2 型文法または文脈自由 context free 文法であると言われる。$A \to \beta$ という生成規則は，A がどのような文脈に現れるかに無関係に，A を β に置き換えることを許すものである。そのため文脈自由という呼び名が生まれた。

2.4 3 型文法

文脈自由文法をさらに制約して

$$A \rightarrow \alpha B \ (A, B \in V_N) \tag{7}$$

$$A \rightarrow \alpha \ (\alpha \in V_T) \tag{8}$$

と制限した場合を **3 型文法**, あるいは**正規文法** と呼ぶ。すなわち A, B ともに非終端記号の集合 V_N の要素であり，α は終端記号である場合である。

ある文法 $G = (V_N, V_T, P, S)$ において生成規則 P が $A \rightarrow aB$, または $A \rightarrow a$ という形であるとする。ここで A, B は変数を，a は終端記号をあらわす。このような文法 G は 3 型または正規 regular 文法と呼ばれる。

2.5 それぞれの文法の能力

定義から明らかにどの正規文法も文脈自由であり，どの文脈自由文法も文脈依存であり，どの文脈依存文法も 0 型である。文脈依存 context sensitive 文法を csg, 文脈自由 context free 文法を cfg, 正規 regular 文法を rg と略記する。0 型言語は recursively enumerable(再帰的に可算) r.e. 集合と略記される。

0 型文法から 3 型文法までのそれぞれの文法は本質的に異なった能力を持っているとされる。図に示すと図 1 のようになる。ここで，RL は正規言

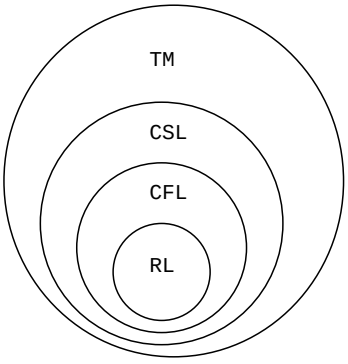

図 1: 形式言語相互間の関係

語，CFL は文脈自由形言語，CSL は文脈規定型言語，TM はチューリング機械である。

3 型言語を受理するのは有限オートマトンであり，2 型言語を受理するのは**プッシュダウンオートマトン pushdown automaton** と呼ばれる。1 型言語を受理するモデルは**線形拘束オートマトン linear bounded automaton**

であり，0 型言語を受理するモデルは**チューリング機械 Turing machine** と呼ばれている。

句構造文法階層	文法	言語	オートマトン	生成規則
0 型		帰納的可算	チューリングマシン	制限なし
1 型	文脈依存	文脈依存	線形拘束オートマトン	$\alpha A\beta \to \alpha\gamma\beta$
2 型	文脈自由	文脈自由	プッシュダウンオートマトン	$A \to \gamma$
3 型	正規	正規	有限オートマトン	$A \to a$ および $A \to aB$ または $A \to Ba$

3　有限オートマトン

先に示したとおり，言語の持っている情報を条件付き確率で表したのがマルコフ・モデルである。マルコフモデルは，文の生じやすさを確率を使って表現したものである。ここである言語において発生しうるものは全て同等に扱うとい立場に立てば，マルコフモデルにおける確率の代わりに，可能か否かの 2 値の確率をもったモデルを作ることができる。このような場合のモデルを**有限オートマトンモデル finite automaton model:fa** と呼ぶ。有限オートマトンは文法によって定義されるすべての言語を定義することはできない。有限オートマトンによって定義しうる言語は正規言語 regular language と呼ばれる。

有限オートマトンモデルで，ある与えられた文字列がその言語の文となっているかどうかを検討する場合，その文字列を**初期状態 initail state** に対して与えると，入力されてくる各文字とシステムの内部状態とで，次に移るべき状態が決定され，有限オートマトンの**状態推移 state transition** が起こる。そして，全ての文字列を入力し終えた時点で，この有限オートマトンがあらかじめ定められている**最終状態 final state** となっている場合に，この文書は**受理 accept** されたという。

言語の形式的モデルにはいくつかある。そのうち有限オートマトン理論は最も単純なものであり，マルコフモデルの延長上にあると考えて良い (**??**)。

3.1　有限オートマトン理論の基礎概念

オートマトンの定義は以下のように与えられる

1. 状態の集合を $K = \{q_0, q_1, \cdots, q_n\}$ とする。q_i は状態で，n は有限である。

2. 入力文字の集合を S とする。S は有限である。

3. 状態推移関数の集合を P とする。P は有限である。

4. 最終状態の集合を F とする。$F \subset K$ である。

このような有限オートマトンを

$$M = (K, V_T, \delta, S, F) \tag{9}$$

と書き表す。あるオートマトン M がある状態 q_i にあるとき，入力 α が与えられたら状態が q_j に変化したとする。この場合

$$\delta(q_i, \alpha) = q_j \tag{10}$$

と書き，δ を状態遷移関数と呼ぶ。

文 x は $\delta(q_0, x) = p$ なる p が最終状態 F に属するとき，オートマトン M によて受理 accept されたという。オートマトン M によって受理されるすべての文 x の集合を $T(M)$ であらわす。すなわち，

$$T(M) = \{x | \delta(q_0, x) \text{ は } F \text{ に属する } \} \tag{11}$$

有限オートマトンによって受理される列の集合は，正規 である。

アルファベット S 上で文字列を定義する。これは，S の文字を任意の回数だけ繰り返して並べたものである。アルファベット 1 文字の集合を S とし，2 文字からなる集合を S^2, n 個ならべたものを S^n とする。特別なものとして，文字を 0 個並べたものを

$$S^0 = \{\lambda\} \tag{12}$$

と表現する。λ を**空語（空列）null string** という。すべての集合の和集合

$$S^* = S^0 + S^1 + S^2 + \cdots + S^n + \cdots = \sum_{i=0}^{\infty} S^i \tag{13}$$

$$S^+ = S^1 + S^2 + \cdots + S^n + \cdots = \sum_{i=1}^{\infty} S^i \tag{14}$$

と表す。S^* をアルファベットの**閉包 closure** と呼ぶ。すなわちアルファベットの文字を任意の個数（0 も含んで）だけ並べた**文字列 character string** 全体の集合である。たとえば $S = \{a, b, c\}$ とすると，$S^0 = \{\lambda\}$, $S^1 = \{a, b, c\}$, $S^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$, $S^3 = \{aaa, aab, aac, aba, \cdots\}$, などとなる。アルファベット S を $A \sim Z$, $a \sim z$, $0 \sim 9$, ピリオドやスラッシュなどの記号などにとれば，S^* には英語のすべての単語，すべての文，すべての文章，になる。S^* の中にはシェイクスピアの作品から，これまでに書かれた，そしてこれから書かれるであろうすべての文書を含む集合である。

S^* はアルファベット S のすべての文字列を含んでいるため，特別な興味はない。S^* の中のある文字列 α を取り出してみると，英語の文になってい

ることもあれば，でたらめな文字列であることもある。ある文字列 α が言語 L の文となっている場合，これを

$$\alpha \in L \tag{15}$$

と書き，そのようなすべての α の集合を**言語** L と定義する。当然ながら，

$$L \subset S^* \tag{16}$$

L は S^* のサブセットであるという。

3.2 有限オートマトン

有限オートマトン M において，

$$M = (K, V_T, \delta, S, F) \tag{17}$$

アルファベット V_T 上の有限オートマトン M とは次のような体系 (K, V_T, δ, q_0, F) のことである。ここに K は状態 state の空でない有限集合，V_T は有限の入力アルファベット input alphabet, δ は $K \times \Sigma$[1] から K の中への関数。q_0 は初期状態 initial state で K に属し，F は最終状態 final state の集合で $F \subseteq K$ である。

有限オートマトンモデルで，ある与えられた文字列がある言語の文となっているかどうかを検定したいという場合を考える。その文字列をモデルの**初期状態 initial state** に対して与えると，入ってくる文字列の各文字と内部状態の組み合わせで，次に移るべき状態が決定され，有限オートマトンの**状態推移 state trasition** が行われる。文字列を最後まで入れ終わった状態でオートマトンがあらかじめ定められている**最終状態 final state** となっている場合，この文字列は**受理 accept** されたという。このようにして文字列がその言語に属するかどうかが判別できる。このようなオートマトンは次のように定式化でき，

1. 状態の集合を $K = \{q_0, q_1, \cdots, q_n\}$ とする。q_i は状態で，n は有限である。
2. 入力文字列の集合を Σ とする。Σ は有限である。
3. 状態推移関数の集合を δ とする。δ は有限である。
4. 初期状態を q_0 とする。$q_0 \in K$ である。
5. 最終状態の集合を F とする。$F \subset K$ である。

[1] $K \times \Sigma = \{k, \sigma | k \in K \text{ かつ} \sigma \in \Sigma\}$

このような有限オートマトンを

$$M = (K, \Sigma, \delta, q_0, F) \tag{18}$$

と表記する。

文 x は $\delta(q_0, x) = p$ なる p が最終状態の集合 F に属するとき，オートマトン M によって受理されたという。オートマトン M によって受理されるすべての文 x の集合を $T(M)$ であらわす。すなわち，

$$T(M) = \{x | \delta(q_0, x) \text{ は } F \text{ に属する } \} \tag{19}$$

オートマトン M がある状態 q_i にあるとき，入力 a が与えられたら，状態 q_i に移行する場合，

$$\delta(q_i, a) = q_j \tag{20}$$

と書き，**状態推移関数**という。δ はこのような関数の集合である。この状態推移関数に対応して，次の図 2 のように書ける。

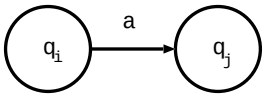

図 2: 状態遷移図

簡単のため，文字 a が任意個並んだ文字を受理する有限オートマトンを作ってみる。これは

$$S^* = \lambda + a + aa + aaa + \cdots + a^n + \cdots \tag{21}$$

という集合を受理するものである。これは図 3 に示すように これは 1 つの状

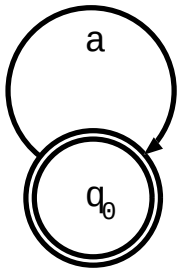

図 3: 状態遷移図その 2

態をもつオートマトンで実現できる。これを詳しく書くと，

$$\begin{array}{lcl} K & = & \{q_0\} \\ \Sigma & = & \{a\} \\ P & = & \{\delta(q_0, a) = q_0\} \\ \text{初期状態} & = & \text{最終状態} = q_0 \end{array} \tag{22}$$

図 4 には，ある有限オートマトンの完全な記述を示した。

$M = (K, \Sigma, \delta, q_0, F)$

$\Sigma = \{0, 1\}$

$K = \{q_0, q_1, q_2, q_3\}$

$F = \{q_0\}$

$\delta(q_0, 0) = q_2,\ \ \delta(q_0, 1) = q_1$

$\delta(q_1, 0) = q_3,\ \ \delta(q_1, 1) = q_0$

$\delta(q_2, 0) = q_0,\ \ \delta(q_2, 1) = q_3$

$\delta(q_3, 0) = q_1,\ \ \delta(q_3, 1) = q_2$

状態図は，状態を表すノードと，状態の変化を表す矢印とから成っている。

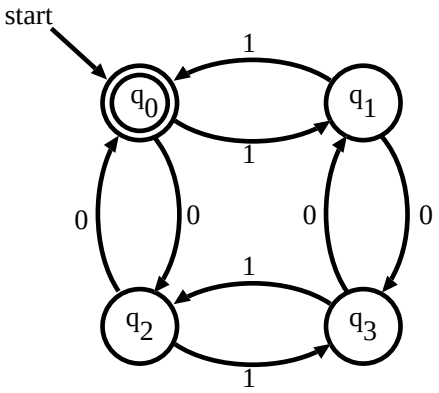

図 4: 偶数個の 0 と偶数個の 1 とをもつ列の集合を受理する有限オートマトンの状態図

すなわち，オートマトンが状態 q で入力記号 a を読んだとき状態 p に移るならば，q から p への矢印を書き，それに標識 a をつける。最終状態，すなわち F に属する状態は 2 重の円で示されている。初期状態は start と書かれた矢印で示されている。

図 4 で M への入力が 110101 であったとする。$\delta(q_0, 1) = q_1$ でしかも $\delta(q_1, 1) = q_0$ であるから，$\delta(q_0, 11) = q_0$. このように $11 \in T(M)$ である。今問題にしているのは 110101 であるが，$\delta(q_0, 0) = q_2$ から $\delta(q_0, 110) = q_2$ を得る。次は $\delta(q_2, 1) = q_3$ から $\delta(q_0, 1101) = q_3$ がわかる。最後に $\delta(q_3, 0) = q_1$ および $\delta(q_1, 1) = q_0$ から $\delta(q_0, 110101) = q_0$　となり，110101 が $T(M)$ に属することがわかる。$T(M)$ が $\{0, 1\}^*$ の文で，偶数個の 0 と偶数個の 1 とを含むものの全体の集合であることは容易に確かめられる。

3.3 非決定性有限オートマトン

非決定性有限オートマトン M とは，状態の空でない有限集合 K, 有限の入力アルファベット Σ, $K \times \Sigma$ の要素を K の部分集合に対応させる関数 δ, K に属する初期状態 q_0, 最終状態の集合 $F \subseteq K$ の組 $(K, \Sigma, \delta, q_0, F)$ のことである。これに対し，前節オートマトンを決定性オートマトンという。

決定的な場合と非決定的な場合の違いは，非決定的なオートマトンが $\delta(q, a)$ が一個の場合ではなく，状態の（空かもしれない）ある集合であるという点にある。$\delta(q, a) = \{p_1, p_2, \cdots, p_k\}$ の解釈は有限オートマトン M が，状態 q で入力を読み込んだとき，その次の状態して $p_1, p_2, \cdots, p_k$ のどれかを選ぶということである。

文 x は，最終状態の集合 F と関数 $\delta(q_0, x)$ との両方に属するような状態 p が存在するとき，オートマトン M に受理されるという。

3.4 正規言語

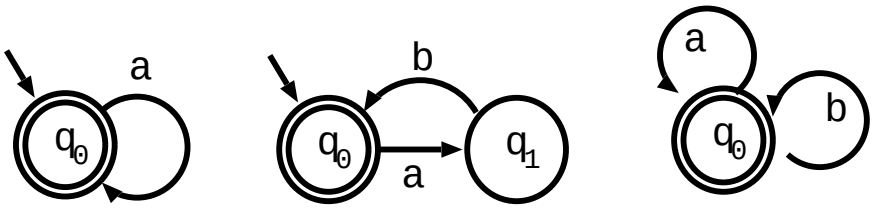

図 5: 3 つのオートマトン

図 5 の左図は

$$\{a^n\} \quad n = 1, 2, \cdots \tag{23}$$

という文字列を受理する。中央の図は同様にして

$$\{\lambda, ab, abab, \cdots, (ab)^n, \cdots\} = \{(ab)^n | n = 0, 1, 2, \cdots\} \tag{24}$$

という文字列を受理する。任意の n について a^n の集合 $\{a^n | n = 0, 1, 2, \cdots\}$ は，a^* と表記することができるから，図 5 の左と中央の有限オートマトンは，それぞれ

$$a^*, \ (ab)^* \tag{25}$$

という文字列を受理することになる。図 5 の右図のオートマトンは，文字 a と b のいずれもが任意の回数と組み合わせを受理できる。これは，

$$(a + b)^* \tag{26}$$

と表現できる。ここで + は a と b のどちらでも良いという意味で論理和 OR の意味である。また,

$$(a+b)^* = \lambda + (a+b) + (a+b)^2 + (a+b)^3 + \cdots \quad (27)$$

$$= \lambda + a + b + aa + ab + ba + bb + aaa + aab + aba + \cdots \quad (28)$$

となり，a と b の任意の組み合わせができたことがわかる。

状態遷移図と受理される文字列の集合との間には、以下のような関係がある。

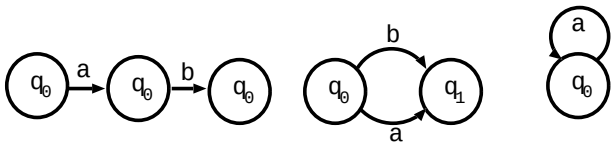

図 6: オートマトンと演算の対応関係

1. 状態遷移を二つ直列に繋いだ状態に対しては，入力文字を二つ連接した文字列が対応する。図 6 左
2. 二つの状態遷移が同じ二つの状態の間に存在する場合には，二つの入力文字の集合が対応する。図 6 中
3. １つの状態から出て，同じ状態に戻ってくるループがある場合，そのループをたどる文字列を x としたとき，文字列 x の閉包 * が対応する。図 6 右

すべての有限オートマトン M はこの 3 つの場合に分解することができる。ゆえに，上記３つの組み合わせを合成することで M が受理する文字列の集合が明らかとなる。(1) の場合は文字列が順序づけられた積の形となり，記号 “・” 記号を用いて連接を表現する。一般の積の概念の拡張であり，記号 “・” は省略が可能である。(2) に場合は和として “+” を用いる。(3) の場合は “*” と表記する。**文字列をこれらの記号によって表現した文字列の集合を正規集合 regular set あるいは正規言語 regular language** と呼ぶ。

図 7 では，状態 q_0 から q_1 へは文字列 $(a+b)$ で推移し，q_0 から q_2 に至るためには

$$(a+b)c(b((a+b)c)^* \quad (29)$$

なる文字列の集合が対応する。最終状態 q_3 に至るためには

$$(a+b)c(b((a+b)c)^*a \quad (30)$$

という集合が対応する。すなわちこのオートマトンが，このようにしてが与えられれば，それが受理する正規言語を得ることができる。換言すれば，有限オートマトンが与えられれば，それが受理する正規言語を得ることができる

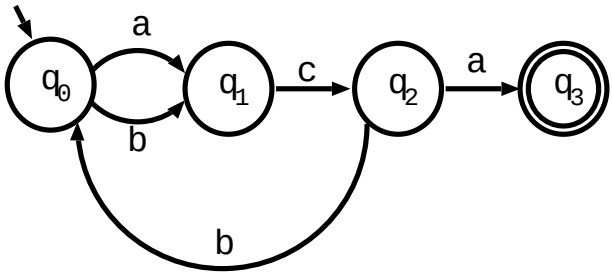

図 7: $(a+b)c(b(a+b)c)^*a$

4 文脈自由型文法

自然言語の文の構造を取り扱うときには，文脈自由型句構造文法を用いる。

4.1 形式文法

有限オートマトンは，閉包の演算によってのみ無限の文を生成することができる。正規言語において，ある文字の影響は有限の長さにしか及ばない。すなわちある言語のアルファベットの文字 a が現れた後，何文字離れて文字 b が生じるかという形の依存関係は，有限オートマトンが持っている内部状態の数以上には及ばない。ということを明白に述べたのがチョムスキーであった。

彼の説明によれば，

$$\{a^n b^n | n = 0, 1, 2, \cdots\} \tag{31}$$

という言語では，任意の n について a が n 個並べは，続いて b が n 個並ぶという意味で n 個先まで依存関係が存在するが，このような任意の n を計数し記憶するためには，有限の記憶しか持たない有限オートマトンでは不可能であることがわかった。

チョムスキーが挙げた例に

$$\text{if}\,(s_1)\,\text{then}\,(s_2) \tag{32}$$

$$\text{either}\,(s_3)\,\text{or}\,(s_4) \tag{33}$$

$$\text{the man who said that}\,(s_5)\,\text{is arriving tody} \tag{34}$$

がある。これらの文の s_1, s_2,$\cdots$,s_5 には任意の長さの文を入れることができるので，それぞれ自分自身の文を何重にも繰り返しいれてみる。たとえば，式 (32) に対して代入を繰り返すと

$$\text{if}(\text{if}(s_1)\text{then}(s_2))\text{then}s_2 \cdots \tag{35}$$

これは結局

$$(\text{if})^n s_1(\text{then}\,(s_2))^n \quad (n = 1, 2, \cdots) \tag{36}$$

という形の文となる。この言語も正規言語ではない。

これを克服したのが，チョムスキーの**文脈自由型句構造文法 context–free phrase structure grammar:CFG** と**文脈規定型句構造文法 context sensitive phrase grammar:CSG** である。

すなわち，文は主語と述語から成り立っており，以下同様に句を分解して行く。文の構造を全体から細部に向かっての句の形で順次処理していく。その中心は**句構造規則（書き換え規則）**にある。以下に例を載せる。

$$\begin{aligned}
\text{文} &\rightarrow \text{主語}\quad\text{述語}\ (\text{S}\rightarrow \text{SUB PRED}) && (37)\\
\text{主語} &\rightarrow \text{固有名詞}\ (\text{SUB}\rightarrow \text{PROPN}) && (38)\\
\text{主語} &\rightarrow \text{名詞句}\ (\text{SUB}\rightarrow \text{NP}) && (39)\\
\text{名詞句} &\rightarrow \text{名詞}\quad(\text{NP}\rightarrow\text{N}) && (40)\\
\text{名詞句} &\rightarrow \text{冠詞 名詞}\ (\text{NP}\rightarrow\text{DET N}) && (41)\\
\text{名詞句} &\rightarrow \text{冠詞 形容詞 名詞}\ (\text{NP}\rightarrow\text{DET ADJ N}) && (42)\\
\text{述語} &\rightarrow \text{自動詞}\ (\text{PRED}\rightarrow\text{V1}) && (43)\\
\text{述語} &\rightarrow \text{自動詞 前置詞句}\ (\text{PRED}\rightarrow\text{VT PP}) && (44)\\
\text{述語} &\rightarrow \text{他動詞}\quad\text{名詞句}\ (\text{PRED}\rightarrow\text{VT NP}) && (45)\\
\text{前置詞句} &\rightarrow \text{前置詞}\quad\text{名詞句}\ (\text{PP}\rightarrow\text{PREP NP}) && (46)\\
\text{固有名詞} &\rightarrow \text{Tom, John, Shin,}\cdots && (47)\\
\text{名詞} &\rightarrow \text{piano, desk, boy,}\cdots && (48)\\
\text{動詞} &\rightarrow \text{plays, is, hit,}\cdots && (49)\\
\text{形容詞} &\rightarrow \text{beautiful, healthy,}\cdots && (50)\\
\text{冠詞} &\rightarrow \text{a, the, }\cdots && (51)
\end{aligned}$$

ここに矢印の左辺に表れる記号を**非終端記号 non-terminal symbol** といい，その集合を V_k で表す。矢印の右辺にしか現れない**非終端記号 terminal symbol** といい，その集合を V_T で表す。

このような文法の規則は**生成規則 generation rule** または**書き換え規則 rewriting rule** と呼ぶ。たとえば上の第一行目は「文は主語と述語とからなる」と読むことができる。ある文 S から出発して順次書き換え規則を適用して文を作り出すことができる。

$$\begin{aligned}
S &\Rightarrow \text{SUB PRED} \Rightarrow \text{PROPN PRED} && (52)\\
&\Rightarrow \text{PROPN VT NP} && (53)\\
&\Rightarrow \text{PROPN VT DET N} && (54)\\
&\Rightarrow \cdots \Rightarrow \text{Tom play the piano} && (55)
\end{aligned}$$

$\Rightarrow$ は左辺から右辺が**導出 derive** されると読む。導出の繰り返しを $\overset{*}{\Rightarrow}$ で表現し，

$$S \overset{*}{\Rightarrow} \text{Tom plays the piano.} \tag{56}$$

などと書く。すなわち非終端記号に書き換え規則を順次適用していって，書き換える規則がなくなったとき，文が生成されたと考えるのである。これがチョムスキーが提唱した生成文法 generative grammar の本質であった（この辺りは前回までの復習になる）。

しかし，このような単純な書き換えだけでは十分に説明されない言語現象もあることがチョムスキーによって指摘された。Aux を助動詞として

$$NP_1 \ \text{Aux} \ \text{VT} \ \ NP_2 \tag{57}$$

が文である。この受動態は

$$NP_2 \ \text{Aux be VTed by} \ NP_1 \tag{58}$$

であるがこのような変形規則を CSG, CFG は表せない。

4.2 文脈自由型句構造言語

非終端記号の集合を V_N, 終端記号の集合を V_T として次の規則

$$A \to \alpha \ \ (A \in V_N, \alpha \in (V_N \cup V_T)^+) \tag{59}$$

からなる生成規則の集合 P があるとする。書記記号 $\sigma \in V_N$ があって文法は

$$G = (V_N, V_T, P, \sigma) \tag{60}$$

と表現される。

たとえば文法 G が

$$G = (\{\sigma\}, \{a, b\}, P, \sigma) \tag{61}$$

$$P : \sigma \to \sigma a \sigma, \ \ \sigma \to b \tag{62}$$

において

5 プッシュダウンオートマトン

正規言語を受理する機構は有限オートマトンであった。文脈自由文法を受理する機構はプッシュダウンオートマトンと呼ばれるメカニズムである。プッシュダウンオートマトンは図 8 のような構造を持っている。入力テープには受理すべき文字列（文）が記入されていて，左から右に向かって一文字ずつ順に走査入力される。プッシュダウンテープはその右端に z_0 という特殊記号

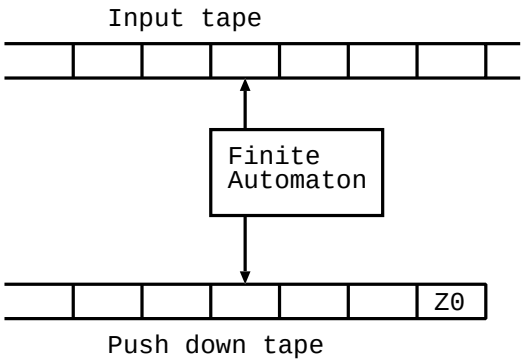

図 8: プッシュダウンオートマトン

が記入されていて，新しく記憶すべき文字はプッシュダウンテープの z_0 の左から順次左に向かって書き込まれていく。このテープ上の文字は消去することができるのであるが，テープに文字を書き込んだ入り，文字を消したりするためのテープヘッドは常にテープに書かれた文字列の最左端にあり，新たに書き込まれる文字はその左につめて書き，消去される文字は必ずテープの最左端である。一文字書き込むとテープヘッドは左に 1 文字移動し，一文字消去するとヘッドは右に移動する。このテープは左側に無限の記憶セルを持っている。このような性質をもつ一時記憶装置をプッシュダウンスタックという。