数理解析研究所講究録
906 巻 1995 年 264-271

自動翻訳における新しい木構造の導入
— 左右木について —

黒川浩一 (Kouichi Kurokawa)　　笠井琢美 (Takumi Kasai)

平成 7 年 3 月 10 日

Abstract

本稿は、著者らが開発した英和自動翻訳システムに関するものである。そのシステムの英文解析機構は、範疇型による英文の解析と、それに伴い導入される新しい木構造である左右木により特徴づけられる。範疇型とは品詞の下位範疇化を進めて符号化し、統語規則としての能力を与えたものであり、左右木とは子供に、右の子と左の子という区別を与えた木構造である本稿では、それらと処理系の形式的定義を与え、それと並行してシステムの能力を紹介する。

1　自動翻訳における木構造

本稿で示される左右木とは、電気通信大学笠井研究室で研究が進められている、英和自動翻訳システム（以下では単にシステムと呼ぶ）において、英文の構造の表現や解析に用いるために導入された新しい木構造である。もちろん従来の生成文法系の文法論による翻訳システムにおいても、木構造が用いられていた。しかしこのとき、その英文の構造を表す木（以下では構文解析木と呼ぶ）のラベルは、葉に英単語が与えられ、それ以外の節点には抽象度の高い「品詞」に相当するもの（以下では構文範疇と呼ぶ）が置かれていた。一方左右木による構文解析木では、すべての節点のラベルが英単語であり、構文範疇が示す情報を、各単語間の位置関係で示している。構文解析木の違いによる利点は [1] で述べた。例文を示そう。

例文 1：The barrister urged the judge to be merciful.

（訳：弁護士は判事に寛大であるように熱心に説いた。[1]）

図 1(a) 従来の生成文法系の文法論による例文 1 の構文解析木

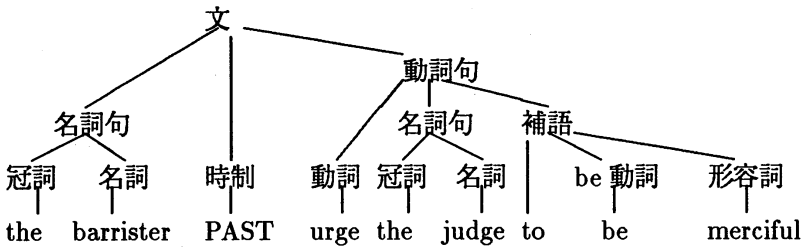
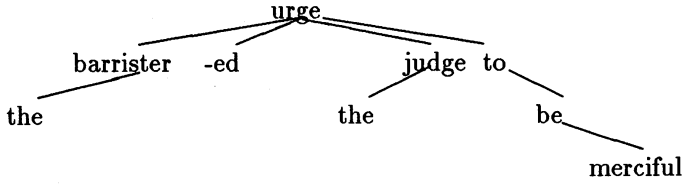


図 1(b) 左右木による例文 1 の構文解析木



[1] 以下本稿で示される訳はすべて本システムにより出力されたものである。

ここで、図 1(a) における PAST や、図 2(b) における -ed は、動詞の過去形を表す記号であり、形態素と呼ばれる。翻訳システムでは単語の解析時において、動詞の過去形や名詞の複数形などを原型とこのような記号に分解して扱っているが、本稿ではこのことはこれ以上扱わない。

図 1(a), 図 1(b) を比較すると、図 1(b) の方が節点数も少なく、構文解析木としてより単純であることが見てとれる。これは、左右木による構文解析木が直接に単語の依存関係を示しており、抽象的な構文範疇をラベルとする節点を持たないためである。このことは、システムの実現における計算量的観点から見て有利である。また言語学者の間には、図 1(a) のような、抽象的な中間節点を持つ構文解析木の心理的実在性を疑問視する向きもある。

図 1(b) のような木から英文を取り出すためには、各節点のラベルを左から右への順に線型に並べればよい。(このような処理をプレスと呼び、後に定義する。) また、この木での各節点間の位置関係はそのラベルの単語間の依存関係を表現している。このため、節点間の位置関係を明確にする必要がある。特に、前から修飾されているのか、後ろから修飾されているのかを区別することは重要である。これらの目的のため、左右木を以下のように定義する。

定義 1．1（結合価付アルファベット）

結合価付アルファベットを、対 (Σ, v) のことと定める。ここでΣはアルファベット、vは、$\Sigma \rightarrow \mathbf{N} \times \mathbf{N}$ なる関数である。vのことを **結合価関数** と呼ぶ。以下では、(Σ, v) のことを単にΣと書く。

$\sigma \in \Sigma$に対し、$v(\sigma) = (l, r)$ であるとき、lをσの**左結合価**、rをσの**右結合価**と呼ぶ。また、$v^{-1}((l, r))$ を $\Sigma_{l,r}$ と書く。

通常のアルファベットΩに対し、関数$d_W : \Omega \rightarrow 2^{\Sigma}$で、$\Omega$ の任意の元 a,b に対し、$[a \neq b \quad iff \quad d_W(a) \cap d_W(b) = \phi]$ を満たすものを考える。この d_W を**単語辞書関数**と呼ぶ。以下では、Ω 上の語の代わりに Σ 上の語、木構造を考える。

(例 1．1)

Ωを英単語全体からなる集合とする。Σ として翻訳システムで用いる疑似的な単語を与える。単語 work $\in \Omega$ に対し、$d_W(work) = \{work_1, work_2, \ldots, xork_k\}$ とする。このとき、$work_1$ は名詞で、$v(work_1)) = (0, 0)$ 。また、$work_i$ $(1 <\leq k)$ は動詞で、それぞれ (1,1),(1,0) などの結合価を持つと解釈できる。ここで、動詞の場合の左結合価は主語を持つので 1 である。一方、右結合価は構文上支配する句（これを**項**と呼ぶ）の数であり、完全自動詞ならば 0 、その他 1,2 が与えられる。

定義 1．2（左右木の台集合）

以下を満たす $(\mathbf{Z} - \{0\})^*$の有限部分集合 D を、**左右木の台集合**と呼ぶ。

1. 任意の $u \in D$ と任意の $u_1, u_2 \in (\mathbf{Z} - \{0\})^*$ に対し、$u = u_1 u_2$ と書けるならば $u_1 \in D$ である。
2. 任意の $u \in D$ と任意の負の整数 i に対し、$ui \in D$ ならば、　$i \leq j < 0$ なる任意の整数 j に対して、$uj \in D$ 。この uj を、u の j **番目の左の子**と呼ぶ。
3. 任意の $u \in D$ と任意の正の整数 i に対し、$ui \in D$ ならば、　$0 < j \leq i$ なる任意の整数 j に対して、$uj \in D$ 。この uj を、u の j **番目の右の子**と呼ぶ。

定義 1．3（左右木）

以下を満たす関数$\alpha : D \rightarrow \Sigma$を$\Sigma$**上の左右木**と呼ぶ。

1. D は左右木の台集合。D の各元を左右木 α の**節点**と呼ぶ。特に、$\varepsilon \in D$ を左右木 α の**根**と呼ぶ。

2. α の任意の節点 u に対し、記号 $\alpha(u) \in \Sigma$ の値の左右の結合価は、その節点の左右それぞれ子の最大数に等しい。すなわち、

$$\forall u \in D, v(\alpha(u)) = (\max(\{|i| \mid ui \in D, i < 0\} \cup \{0\}),\ \max(\{i \mid ui \in D, i > 0\} \cup \{0\}))$$

$\max\{|u||u \in D\} - 1$ を左右木 α の高さと呼ぶ。Σ 上の左右木全体からなる集合を S_Σ と書く。

図 1(b) のような形で左右木を表現する場合には、最内側に 1 番目の節点を書き、外に向かって番号の大きい子供を書くようにする。

定義 1．4（部分木・置換）

Σ 上の左右木 $\alpha \in S_\Sigma$ とある節点 $u \in D$ に対し、集合 $D_u = \{u' | uu' \in D\}$ を考える。集合 D_u は左右木の台集合である。このとき関数 $\alpha_u : D \to \Sigma$ で $\alpha_u(u') = \alpha(uu')$ を左右木 α の節点 u における**部分木**と呼ぶ。

α_1 と α_2 を Σ 上の左右木、D_{α_1} と D_{α_2} を各々の台集合とする。α の節点 $u \in D_{\alpha_1}$ に対し、集合 $D' = (D_{\alpha_1} - \{uu' | uu' \in D_{\alpha_1}\}) \cup \{uu' | u' \in D_{\alpha_2}\}$ を考える。D' は左右木の台集合であり、このとき関数 $\alpha' : D' \to \Sigma$ を任意の $w \in D'$ に対し、

$$\alpha'(w) = \begin{cases} \alpha_1(w) & (w \in D_{\alpha_1} - \{uu' | uu' \in D_{\alpha_1}\}) \\ \alpha_2(u) & (w = uu',\ u' \in D_{\alpha_2}) \end{cases}$$

と定義する。この α' を、左右木 α_1 の節点 u における部分木 α_u を左右木 α_2 で **置換** した木と呼び、$\alpha_1(u \leftarrow \alpha_2)$ と書く。

Σ 上の左右木の、Σ 中の記号と (,) による表現法を以下のように帰納的に定義する。左右木 $\alpha : D \to \Sigma$ に対し、

1. $D = \{\varepsilon\}$ ならば、$\alpha(\varepsilon) = \lambda \in \Sigma_{0,0}$ である、このとき α を λ で表す。
2. $D \neq \{\varepsilon\}$ ならば、$\alpha(\varepsilon) = \sigma \in \Sigma_{l,r}$ である。節点 $-1, \ldots -l, 1, \ldots r$ における α の部分木が各々 $s_{-1}, \ldots, s_{-l}, s_1, \ldots, s_r$ で表現できるとき、α を $(s_{-l} \cdots s_{-1} \sigma s_1 \cdots s_r)$ で表す。

定義 1．5（プレス）

関数 $P : S_\Sigma \to \Sigma^*$ を以下で定義する。Σ 上の任意の左右木 $\alpha \in S_\Sigma$ に対し、

$$P(\alpha) = \begin{cases} \lambda & (\alpha = \lambda \in \Sigma_{0,0}) \\ P(\alpha_{-l}) \cdots P(\alpha_{-1}) \sigma P(\alpha_1) \cdots P(\alpha_r) & (\alpha = (\alpha_{-l} \cdots \alpha_{-1} \sigma \alpha_1 \cdots \alpha_r),\ \sigma \in \Sigma_{l,r}) \end{cases}$$

この関数 P を **プレス関数** と呼び、$P(\alpha)$ を左右木α が表す語と呼ぶ。P は容易に $S_\Sigma \to \Sigma^*$ へと拡張できる。

例 1．2

図 1(b) の左右木は、$\Sigma =$ { the barrister -ed urge the judge to be merciful } 上の左右木である。この場合、単語 urge の結合価は (2,2) であり、merciful の結合価は (0,0) である。urge の左の子は 1 番目が barrister 、2 番目が -ed であり、右の子は、1 番目が judge 、2 番目が to である。merciful をラベルとして持つ節点は、$2 \cdot 1 \cdot 1$ 、judge の下の the をラベルとして持つ節点は $1 \cdot -1$ である。この木をプレスすることにより、列 the barrister -ed urge the judge to be merciful が得られる。

[4],[3] で与えられているような木オートマトンは、容易に左右木に対応するように修正できる。

定義1．6（ボトムアップ左右木オートマトン）

ボトムアップ左右木オートマトンを4組 $M=(Q,\Sigma,f,Q')$ のことと定める。ここで、Q は有限集合で、その元を **状態** と呼ぶ。Σ は入力結合価付アルファベット、$Q'\subseteq Q$ は**最終状態**の有限集合である。また、f は $\Sigma\rightarrow(Q^l\times Q^r\rightarrow Q)$ なる関数で以下を満たすものである。

任意の $\sigma\in\Sigma_{l,r},(l,r>0)$ に対し、$f(\sigma):Q^l\times Q^r\rightarrow Q$ 。

任意の $\sigma\in\Sigma_{0,r},(r>0)$ に対し、$f(\sigma):Q^r\rightarrow Q$。同様に任意の $\sigma\in\Sigma_{l,0},(l>0)$ に対し、$f(\sigma):Q^l\rightarrow Q$。

任意の $\lambda\in\Sigma_{0,0}$ に対し、$f(\lambda)$ は Q のある固定された元。

$\sigma\in\Sigma$に対し、$f(\sigma)$ を f_σ と書く。M において、以下を満たす関数 $\rho:S_\Sigma\rightarrow Q$ を M の**応答関数**と呼ぶ。

1. $\alpha=\lambda\in\Sigma_{0,0}$ ならば、$\rho(\alpha)=f_\lambda$。
2. $\alpha=(\alpha_{-l}\cdots\alpha_{-1}\sigma\alpha_1\cdots\alpha_r),\sigma\in\Sigma_{l,r}$ ならば、$\rho(\alpha)=f_\sigma(\rho(\alpha_{-l}),\cdots,\rho(\alpha_{-1}),\rho(\alpha_1),\cdots,\rho(\alpha_r))$

M によって**受理される木言語**を集合 $S(M)=\{\alpha\in S_\Sigma|\rho(\alpha)\in Q'\}$ と定義する。これにより、M により**受理される言語** を集合 $L(M)=P(S(M))$ と定義する。

2　木範疇文法

前節では左右木を定義した。またすでに述べたように、本翻訳システムでは構文解析木の各節点の位置関係がそれらの節点のラベルとなる単語間の依存関係を表している。本節ではそのような性質を持つ左右木を構成する文法を提案する。

従来の、列に対する生成文法の導出木を基礎とする構文解析木を作成を考える。これは、規則の数を増やすとその選択により計算量が爆発的に増大し、少なくすると英文に対するきめの細かい対応ができなかった。このためある程度の規則で構文解析木をつくり、その後辞書を引いて適当な解析木になっているかを検査する、ということが行われていた。

本システムでは、単語の構文上の出現条件を符号化した範疇型と呼ばれる表現式を導入し、各単語に辞書項目として割り当てる。そして範疇型の演算という形で左右木を成長させて構文解析木を構成していく。本節では範疇型の形式的定義を与え、それを用いた左右木の生成系である木範疇文法を提案する。

定義2．1（範疇型）

A を有限集合とする。次で定義される集合 T の元を**範疇型**と呼ぶ。範疇型は結合価を持つ。

1. $A\subseteq T$ とする。各 A の元を**原子型**と呼ぶ。原子型の結合価は $(0,0)$ とする。
2. 任意の $a\in A$ と任意の $t\in T$ で $v(t)=(l,r)$ なるものに対し $(t/a)\in T$, $(a\backslash t)\in T$ とする。また各々の結合価を $v((t/a))=(l,r+1)$, $v((a\backslash t))=(l+1,r)$ と定める。

また、任意の $a,b\in A$ と任意の $t\in T$ に対し、$(b\backslash(t/a))=((b\backslash t)/a)$ とする。以下範疇型を単に型と呼ぶ。型の二項演算 $\cdot$ を以下で定義する。

$$\forall t,s\in T\qquad(t/s)\cdot s=t\quad s\cdot(s\backslash t)=t$$

この演算を**還元**と呼ぶ。また、T に特別な元 $e,v(e)=(0,0)$ を追加し、任意の $t\in T$ に対し、$e\cdot t=t\cdot e=t$ とする。

Σ から T の有限部分集合への関数 d_T で結合価を保存するものを、Σ 上の**型辞書関数** と呼ぶ。

定義２．２（変数を持つ左右木）

x を結合価 $(0,0)$ を持つ新しい記号とする。以下で定義される集合 S_Σ^x を**変数 x を持つ Σ 上の左右木の集合**と呼ぶ。

1. $S_\Sigma \subset S_\Sigma^x$,　$x \in S_\Sigma^x$
2. 任意の $\sigma \in \Sigma_{l,r}$ に対し、$(x_{-l} \ldots x_{-1}\sigma x_1 \ldots x_r) \in S_\Sigma^x$ ただし、$x_{-l}, \ldots, x_{-1}, x_1, \ldots x_r$ は次の各条件を満たす。
   - ある $0 \leq i \leq r$ なる i が存在して、$x_k \in S_\Sigma, (0 < k \leq i)$ かつ $x_k, (i < k \leq r) = x$ 。
   - ある $-l \leq i \leq 0$ なる i が存在して、$x_k \in S_\Sigma, (i \leq k < 0)$ かつ $x_k, (-l \leq k < i) = x$ 。

   また、$x_{-l} = \ldots = x_{-1} = x_1 = \ldots x_r = x$ であるとき $(x_{-l} \ldots x_{-1}\sigma x_1 \ldots x_r)$ を $\hat{\sigma}$ と書く。

上の定義により、変数を持つ左右木では変数は根の直接の子にしか現れない。しかも根の直接の子は内側から S_Σ の元である、変数を含まない左右木が並び、ある子より外側の子はみな変数 x になっている。いま変数を持つ Σ 上の左右木 $\alpha \in S_\Sigma^x$ と、Σ の記号だけからなる左右木 $\beta \in S_\Sigma$ を考える。α の根の右の子で、変数をラベルとして持つ最も内側の節点を i とする。このとき節点 i を左右木 β で置換して得られる左右木 $\alpha(i \leftarrow \beta)$ は変数を持つ左右木である。この左右木を $\alpha \leftarrow^r \beta$ と書く。同様のことを左の最も内側の変数をラベルとして持つ節点について行なった場合も結果は変数を持つ左右木になり、これを $\alpha \leftarrow^l \beta$ と書く。

このとき、型辞書関数 d_T の定義域を以下のように S_Σ^x に拡張する。$\alpha \in S_\Sigma^x$ に対し、

1. $\alpha = \lambda \in \Sigma_{0,0}$ ならば、$d_T(\alpha) = d_T(\lambda)$,　$\alpha = x$ ならば、$d_T = \{e\}$ 。
2. $\alpha = (\alpha_{-l} \ldots \alpha_{-1}\sigma\alpha_1 \ldots \alpha_r), \sigma \in \Sigma_{l,r}$ ならば、
   $d_T(\alpha) = \{\ t \mid \exists t_i \in d_T(\alpha_i), (-l \leq i < 0, 0 < i \leq r)\ ,\ t' \in d_T(\sigma)\quad t = t_{-l} \cdots t_{-1} \cdot t' \cdot t_1 \cdots t_r\}$

これを用いて、S_Σ^x 上の二項演算 $\odot$ を次で定義する。任意の左右木 $\alpha, \beta \in S_\Sigma^x$ に対し、

1. $(s\backslash t) \in d_T(\alpha)$, $s \in d_T(\beta)$ ならば、$\gamma = \beta \odot \alpha = \alpha \leftarrow^l \beta$。このとき $t \in d_T(\gamma)$ である。
2. $(t/s) \in d_T(\alpha)$, $s \in d_T(\beta)$ ならば、$\gamma = \alpha \odot \beta = \alpha \leftarrow^r \beta$。このとき $t \in d_T(\gamma)$ である。

定義２．３（木範疇文法）

木範疇文法を 4 組 $G = (\Sigma, A, d_T, T_f)$ のことと定める。ここで、Σ は結合価付アルファベット、A は原子型の有限集合、d_T は Σ 上の型辞書関数、T_f は A の有限部分集合であり、その元を**最終型**と呼ぶ。集合 $S(G) = \{\alpha | \alpha \in S_\Sigma, d_T(\alpha) \cap T_f \neq \phi\}$ を G **によって生成される木言語**と呼ぶ。集合 $L(G) = P(S(G)$ を G **によって生成される言語**と呼ぶ。このとき $L(G) = \{w \in \Sigma^* | w = a_1 \ldots a_k, a_i \in \Sigma\ (1 \leq i \leq k)\quad d_T(\hat{a_1} \odot \cdots \odot \hat{a_k}) \in T_f\}$ と書ける。

例２．１

通常のアルファベット Ω を英単語全体、結合価付アルファベットΣ を翻訳システムで用いる疑似的な単語の集合とする。ラベルとしてΣ の元を持ち、型が定義された左右木を、システムにおける**構文解析木**と呼ぶ。次の例文をもとに、型の割り当てと構文解析木の作成を説明する。

例文２：He admitted his guilt to police.

（訳：彼は警察に罪を認めた。）

まず、システムは動詞 admitted が過去形であるので、それを過去を表す記号と原型に置き換えた上で型の辞書を引く。型辞書関数 d_T の値のうち必要なものだけを並べると、

he	-ed	admit	his	guilt	to	police
np	$\Rightarrow vp$	$vp/np;to$ 句	$\rightarrow np$	np	to 句$/np$	np
(0,0)	(0,0)	(1,2)	(0,0)	(0,0)	(0,1)	(0,0)

となる。なお各型において、一番外側の括弧は煩雑なので省略してある。上がシステムの扱う疑似的な単語（以下では単に単語と呼ぶ）の列であり、中段がその単語に与えられた型、下が各々の結合価である。今、基本型は、$A = \{np, sen, to$ 句$\}$ であり、それぞれ名詞、文、前置詞 to による前置詞句を表している。また、$T_f = \{sen\}$ とする。vp は $(np\backslash sen)$ の略であり動詞を表し、$vp/np;to$ 句 は $((vp/to$ 句$)/np)$ の略記である。

$\Rightarrow vp$、$\rightarrow np$ という型は、矢印の向きにその型を修飾することができることを表す型であるが、本稿ではその詳しい説明は省略する。システム上では、修飾した単語をラベルとした節点の子になる。修飾された側の結合価および型は変化しない。矢印による型を省くと以下のような記号列となる。

he	admit	guilt	to	police
np	$vp/np;to$ 句	np	to 句$/np$	np

このとき、型の定義から、

$$v(\widehat{he} \odot ((\widehat{admit} \odot \widehat{guilt}) \odot (\widehat{to} \odot \widehat{police}))) = \{sen\}$$

であるので、この列はシステムが用いている木範疇文法を G とすると、$L(G)$ の元である。実際には矢印を用いた型を持つ単語も含めた次ぎのような木が生成され、翻訳システムにより受理される。

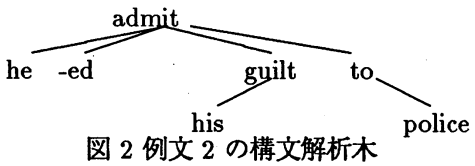


図 2 例文 2 の構文解析木

先に型は構文範疇を下位範疇化して符号化されたものであると述べた。例えばこの例文の場合、動詞 admit に与えられている型 $vp/np;to$ 句 は、正確に書くと $(((np\backslash sen)/to$ 句$)/np)$ である。この型は、右に名詞句と to による前置詞句をこの順で従え、左に名詞句（主語）を持つことによりひとつの文となることができる、という構文上の出現条件を表している。。従って、この動詞 admit の代わりに同様の型を持ち得る動詞、例えば write をおいた場合でも、意味はなさないがひとつの文として解釈できる。一方、このような型を持たない動詞では、構文解析は失敗し、その列はシステムにより拒否される。

3 結合価付プッシュダウンオートマトン

前節で型による左右木の文法を定義した。ここではその文法と等しい列を受理するオートマトンを定義する。

定義 3．1（結合価付プッシュダウンオートマトン）

結合価付プッシュダウンオートマトンを 6 組 $N = (Q, \Sigma, \Gamma, \delta, q_0, \Gamma')$ のことと定める。ここで、Q は、有限集合でその元を状態と呼ぶ。Σ は、入力結合価付アルファベット、Γ はプッシュダウン結合価付アルファベットである。$q_0 \in Q$ は**初期状態**、$\Gamma' \subseteq \Gamma_{0,0}$ は**受理スタック記号**の集合。δ は以下を満たす $(Q \times \Sigma \cup Q \times \Gamma \times \Gamma) \rightarrow 2^{Q \times \Gamma}$ なる関数である。

- 任意の状態 $p, q \in Q$ と $\sigma \in \Sigma$，$\gamma \in \Gamma$ に対し、$(p, \gamma) \in \delta(q, \sigma)$ ならば $v(\sigma) = v(\gamma)$ を満たす。

- 任意の状態 $p,q \in Q$ と $\gamma_1,\gamma_2,\gamma \in \Gamma$ に対し、$(p,\gamma) \in \delta(q,\gamma_1,\gamma_2)$ ならば各プッシュダウンアルファベットの結合価は
  - $v(\gamma_1)=(0,0),\ v(\gamma_2)=(l,r),\ v(\gamma)=(l,r-1)$ または
  - $v(\gamma_1)=(l,r),\ v(\gamma_2)=(0,0),\ v(\gamma)=(l-1,r)$ を満たす。

$Q \times \Sigma^* \times \Gamma^*$ の元を N の**様相**と呼ぶ。様相上の関係 $\Longrightarrow$ を次で定義する。任意の状態 $p,q \in Q$ と列 $w,w' \in \Sigma^*$ およびプッシュダウンアルファベットの列$\phi,\phi' \in \Gamma^*$ に対し、$(q,w,\phi) \Longrightarrow (p,w',\phi')$ であるのは、

- ある $a \in \Sigma$ と $\gamma \in \Gamma$ が存在して $w=aw',\ \phi'=\gamma\phi$ と書けて $(p,\gamma) \in \delta(q,a)$ が成り立つ、または
- $w=w'$ であり、ある $\gamma,\gamma_1,\gamma_2 \in \Gamma$ が存在して $\phi=\gamma_1\gamma_2\phi'',\ \phi'=\gamma\phi''$ と書けて $(p,\gamma) \in \delta(q,\gamma_1,\gamma_2)$ が成り立つ

時であり、かつその時に限る。様相 c_1,c_2 に対し、関係 $c_1 \Longrightarrow c_2$ を、様相 c_1 から様相 c_2 への結合価付プッシュダウンオートマトン N の**動作**と呼ぶ。関係$\Longrightarrow$ の反射推移閉包を $\Longrightarrow^*$ で表わす。

集合 $L(N)=\{w \in \Sigma^* | (q_0,w,\varepsilon) \Longrightarrow^* (q,\varepsilon,\gamma)\ \ q \in Q,\ \gamma \in \Gamma'\}$ を N**が受理する言語**と定義する。

以下の定理がオートマトンの動作ステップ数や木の高さに関する帰納法などで示される。証明は本稿では省略する。

定理

ボトムアップ左右木オートマトンの受理する言語のクラスと、木範疇文法により生成される言語をプレスしたもののクラスの、結合価付プッシュダウンオートマトンにより受理される言語のクラスは互いに等しい。

4 まとめ

本稿では翻訳システムを形式的に表現するための道具立てとして、新しい木構造である左右木を定義し、その文法と、それと能力の等しい機構を定義した。実働モデルである翻訳システムによる例文とその翻訳結果を APPENDIX に示しておく。

結合価付プッシュダウンオートマトンは、容易に左右木を出力する機構を持たせることができる。現に翻訳システムではこのように扱われている。このとき、定理はより強い形で、すなわちボトムアップ左右木オートマトンの受理する木言語のクラスと、木範疇文法の生成する木言語のクラスと、結合価付プッシュダウンオートマトンの出力する木言語のクラスが互いに等しいことが示せるだろう。

左右木による構文木では、すべての節点のラベルが単語であり、その位置関係で単語間の関係を表わしている。これは従来の、言語は列でありその深層構造として構文木をとらえている方法とは明かに異なるものである。

なお例の中で触れたように、本稿で示した範疇型はその本質的部分である/,\ によるもののみであり、実働モデルで用いている矢印を用いた型は扱われていない。このような型に対する形式的な扱いは今後の課題である。

APPENDIX — 翻訳システムによる出力 —

本稿で紹介した翻訳システムにより翻訳された例文の一部を示しておく。

- Dr. Brown is specialist in chest diseases.
  ブラウン博士は胸部疾患の専門家である。
- He began talking about his family.
  彼は家族について話しはじめた。
- He dared me to jump across the stream.
  彼は私にその小川を飛び越せるならやってみろと言った。
- He offered drinks to everyone in the bar.
  彼はバーにいたすべての人に飲み物を提供した。
- John appears to want to be loved by Mary.
  ジョンはメアリーに愛されたいと思っているように見える。
- Miss Nancy is an angel of a nurse.
  ナンシーさんは天使のような看護婦だ。
- She read the letter to all her friends.
  彼女は友達みんなにその手紙を読んで聞かせた。
- He promised me to be here.
  彼は私にここにいると約束した。
- He sold his car to one of his neighbours.
  彼は近所の人に車を売った。
- There's still time for us to see the film
  . まだ私たちが映画を見る時間はある。
- They left me to do all the dirty work.
  彼らは私に汚い仕事をみんなさせた。
- We owe it to society to help in apprehension of criminals.
  私たちは犯人の逮捕に協力するという義務を社会に対し負っている。

参考文献

[1] 黒川浩一・笠井琢美, 英文の型とその演算—英文翻訳システムの試作—, 1994 年夏の LA シンポジウム

[2] ピーター・セルズ　郡司隆男、田窪行則、石川彰 訳, 現代の文法理論, 産業図書,1988

[3] W.C.Rounds,Mapping and Grammars on Trees,*Mathematical Systems Theory 4(1970),257-287*

[4] W.S.Brainerd,Tree Generating Regular Systems,*Information and Control 14(1969),217-231*

[5] J.Lambek,The mathematics of sentence structure,*American Mathematical Monthly 65(1958),154-170*

[6] A.Burghelea,Syntactic types and theory of categories,*REVUE ROUMAINE DE MATHEMATIQUES PURES ET APPLIQUEES TOME XII(1967),363-371*