自然言語処理　37－3
（1983. 5. 13）

言語運用のモデルに基づく実時間の構文解析

橘田浩一，山田尚勇（東京大学理学部）

1．はじめに

ここで紹介する研究の目標は、人間の言語処理の過程を明らかにすることと、その挙動をまねるようなシステムを計算機上に実現することにある。人間の言語処理の外面的な特徴のうち、われわれが注目したのは、それが普通は実時間で行なわれる、という点である。

自然言語を構文解析する手法として、今日までさまざまなものが提案されてきたが、それらの大部分は入力文の長さに関して線型より大きな時間計算量をもっている。たとえば、ATN[1]やDCG[2]などの仮説駆動（hypothesis driven）型のやりかたは、一般に入力文の長さの指数関数にあたる時間計算量をもち、LINGOL[3] などで用いられているデータ駆動（data driven ）型のアルゴリズムは三次多項式の時間計算量をもっている。すなわち、これらの手法は、時間計算量に関して人間の振舞いに十分近いものとは言えない。

従来の方法で実時間の処理が行なえないのは、それらが言語能力（linguistic competence ）の反映である文法の記述に偏し、言語運用（linguistic performance）に関する十分な洞察に基づいていないためであるとわれわれは考える。特に欠けていると思われるのは、人間の短期記憶の有限性[4,5]に関する配慮である。

われわれはまず、言語運用に見られるある側面を短期記憶の有限性と関係づける代数的なモデルを作成した。これは発話のモデルで、人間が発することのできる全ての文を生成する能力を持つ有限オートマトンである。これを変型することによって実時間の構文解析システムを実現できるわけである。ただし、残念ながらこのシステムは人間が言語を認識する過程のモデルとしては十分でない。

ここでの議論は、対象とする自然言語の文法が文脈自由な形に書けることを前提しているが、最近の拡張文脈自由文法（generalized phrase structure grammar）[6,7] についての議論にも見るように、これは文法が変型文法の枠内で記述できるという前提と同程度の一般性を持っているとわれわれは考える。

さらに、われわれの手法は、ほぼ純粋に言語運用論的なものであり、文法に対しては、それが文脈自由な形に書けること以外にいかなる条件をも課さない。したがって、文法の記述は文脈自由である限りここでの議論とは独立に論じられる。特に、構文解析の際に複数個の可能な構文木のうちどれを選ぶかというような問題は、たとえば確率的文脈自由文法[8] の枠組の中で処理するべきもので、われわれの手法が関知するところではないと考えられる。また、われわれの議論は全く構文論的なもので、意味的な処理はここで述べる手法とは独立に扱われる。

なお、以下の議論のうち定理の証明は省略した。詳細については橘田[9]を参照して頂きたい。

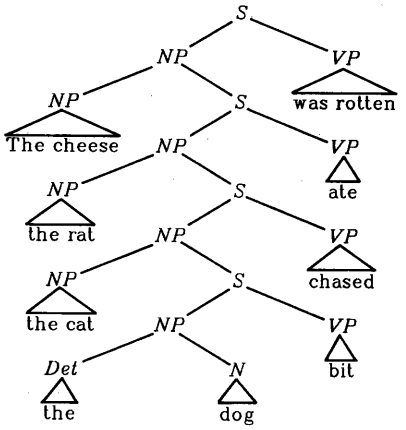
2．準備

ここでは以後の議論に必要な用語を定義する。

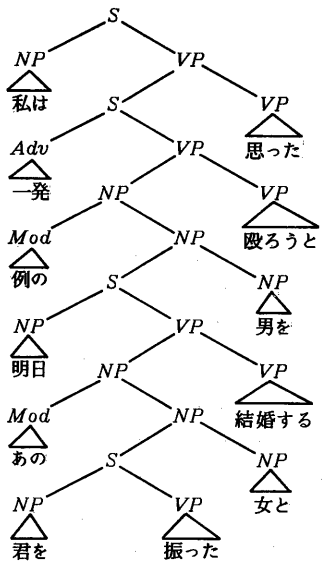
まず、対象とする自然言語をＬと決める。Ｌを記述する文脈自由文法をＧとする。ただし、Ｇが生成する言語Ｌ(Ｇ)がＬに一致するわけではなく、Ｌ(Ｇ)から、実際の運用には用いられない表現を除いたものがＬである。

Ｇ＝＜VN, VT, P, S＞としよう。VNは非終端記号の集合、VTは終端記号の集合、Ｐは生成規則の集合、Ｓ ∈ VNは開始記号である。Ｇはチョムスキー標準形であるとする。つまり、生成規則はＡ→ＢＣまたはＡ→ａという形に限られる。ここで大文字は非終端記号、小文字は終端記号とする。Ａ∈VN, $\alpha \in (VT \cup VN)^2$に対し、(Ａ→$\alpha$)∈Ｐを単にＡ→$\alpha$と書く。

句とは、構文木ないしはその葉の並びの終端記号列のことである。句Ｑが句Ｒの中に入れ子になっているとは、終端記号列α、β が存在し、Ｒ＝αＱβかつ$\alpha\beta$が空列でないことである。このときさらに、αもβも空列でなければ、ＱはＲの中に真の入れ子になっていると言う。句Ｑの（真の）入れ子の深さとは、$0 \leqq i < d$についてＰi+1がＰi の中に（真の）入れ子になっているＰ0 ＝Ｑ，Ｐ1，…，Ｐdが存在するようなｄの最大値のことである。構文木の中の路（path）ｐにおいて、節点Ａが曲がり角であるとは、ＡとＡの親Ｂがｐ上にあり、ＡはＢの左（右）の子であって、Ａの右（左）の子もｐ上にあることを言う。Ｇはチョムスキー標準形であるから、句Ｑが句Ｒの中に真の入れ子になっている条件は、Ｒの根からＱの根までの路が曲がり角を持つことである。



(a)



(b)

図1

3．真の入れ子に関する制約

われわれは理想化された言語能力の反映として文脈自由文法を考えているわけだが、人間の情報処理能力が制限されているために、そのような文法によって生成される「正しい」文でありながら、実際の言語運用には現われないものがある。たとえば、次のような文がそれである。

(a) "The cheese the rat the cat the dog bit chased ate was rotten."

(b)「私は一発例の明日あの君を振った女と結婚する男を殴ろうと思った。」

図1にこれらの文の構造を示す。これを見てすぐにわかるように、どちらの文もかなり大きな真の入れ子の深さを持っている。(a)における真の入れ子の深さは4、(b)においては5である。

一般に、真の入れ子の深さが増すほど、文の生成および認識が人間にとって難しくなる、ということは古くからよく知られており、実際には、真の入れ子の深さがある定数を越えるような表現は使われないと考えられる。たとえば上の例文のようなものは、通常の会話には現われない。

自然な言語運用においては真の入れ子の深さがある定数で抑えられる、という定立を、われわれは真の入れ子に関する制約と呼ぶことにする。以下では、この定立が正しいと仮定した上で話を進めることにする。

真の入れ子によって引き起こされる処理の難しさは、いわゆる garden path 、つまり局所的な構文上の曖昧さによるそれとは異なる。実際、上の例文はgarden pathを含まない。さらに、garden path を含む文が発話されることがしばしばあるのに対し、真の入れ子が深い文は発話されることすらないように見える。

4．短期記憶の有限性

人間の記憶は、いくつかの階層を持っている。それらの階層のうち、短期記憶（short term memory ）あるいは一時記憶（temporary memory）と呼ばれるものは、パターン認識を経て意味づけのなされた情報が初めに入るところであり、かつ言語の発話、認識に伴うような即時的な演算が行なわれる場であると考えられている[5] 。

ところが、この短期記憶には一度に7±2個の情報の塊（chunk ）しか入らない。ただし1個の塊の情報量は可変で、たとえば二進数字の並びを3個づつの組に分け、それぞれの組を0～7の数字だと思って憶えると、普通にそのまま憶えたときの約3倍の長さの並びを正確に憶えることができる[4] 。とは言え、塊は無限に大きくなれるわけではなく、情報の種類と個人ごとに1個の塊が持てる情報量は決まっているようである。

5．言語運用のモデル

ここでは、短期記憶の有限性を満たすことによって真の入れ子に関する制約を説明するような、言語運用の代

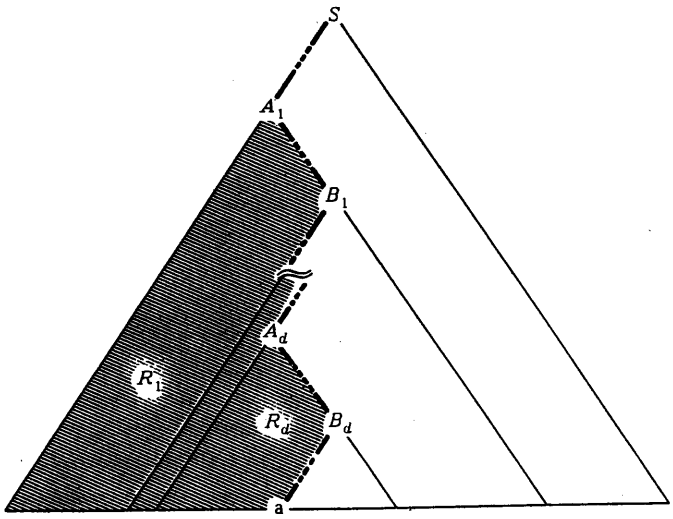


図2

数的なモデルについて述べる。われわれはこのモデルを発話に関するモデルとして提示するわけであるが、そのようなモデルは、人間が発話できるGに関して正しい文であれば生成できる、という条件を満足すべきであり、人間が発話できない文はなるべく生成できない方がよい。発話できるかできないかという規準としてわれわれがここで考えているのは、真の入れ子の深さである。

さて、人間が図2のような構文木を持つ文を発話しており、その葉aの直前までの語列を発声し終えたところであるとしよう。すると、図の斜線を引いた部分の構造はすでに決まっていると考えてよいだろう。これに対して白い部分の構造は話者にとっても未定であり得る。話者は斜線部分と白い部分の文法的な整合を図りつつ発話を続けるものと考えらが、そのために短期記憶内にはいかなる構文的な情報を貯えておけばよいだろうか？

図2のA1，B1，…，Ad，Bdという節点は、斜線部分と白い部分を過不足なく連結している。したがって、これらを短期記憶に入れておけば文全体の文法的な正しさを管理しつつ文を生成できそうである。さらに、根Sから葉aに致る路において、B1，A2，…，Bd-1，Adが曲がり角であることに注意して頂きたい（A1は根Sに一致するかも知れず、またBdはaの親かも知れないので、これらは曲がり角とは限らない。）。したがって、短期記憶の有限性、つまりdの最大値の有限性は、真の入れ子に関する制約に符号すると予想できる。

以上の　議論をもとに、発話のモデルΦ(G，D)を定式化する。Φ(G，D)は有限オートマトンと見なすことができる。その内部状態をsという変数で表わすことにしよう。

$$s \in \bigcup_{i=0}^{2D+1} VN^i$$

である。Φ(G，D)の動きは次の手続きによって定められる。

(1) s：=＜　＞（空リスト）とし、(2)へ行く。

(2) s＝＜A1，B1，…，Ad，Bd＞であるとする。下の(2a)，(2b)　のうち実行可能なものをひとつだけ実行して(3)へ行く。

(2a) A→aなる　　A∈VN，a∈VT が存在するとき、
s：=＜A1，B1，…Ad，Bd，A＞
とし、aを出力する。

(2b) d≧1であり、Bd →aなるa∈VTが存在するとき、
s:=＜A1，B1，…，Ad ＞とし、aを出力する。

(3) s＝＜A1，B1，…，Ad，Bd，C＞であるとする。下の(3a)，(3b)，(3c)　のうち実行可能なものをひとつだけ実行する。

(3a) d＜Dであり、A→CBなるA，B∈VN が存在するとき、
s：=＜A1，B1，…，Ad，Bd，A，B＞
として(2)へ行く。

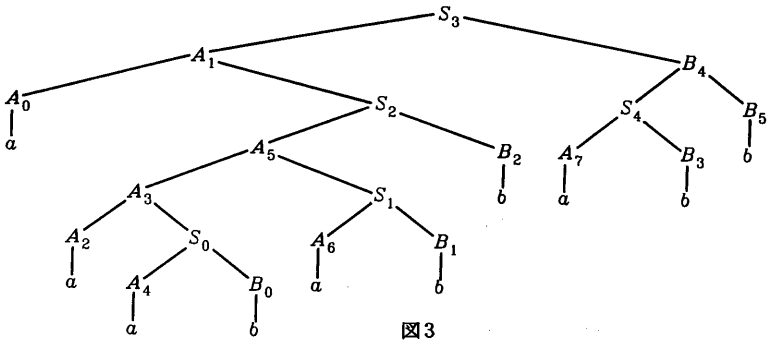


図3

(3b) d≧1であり、Bd→C BなるB∈VN が存在するとき、
s：=<A1, B1, …, Ad, B>
として(2)へ行く。

(3c) s=<S>なら終了する。

図3に示される構文木をもつ文をΦ(G, D)が生成する様子を下に示す。

s の値	出力	規則
< >		
$<A_0>$	a	(2a)
$<A_1, S_2>$		(3a)
$<A_1, S_2, A_2>$	a	(2a)
$<A_1, S_2, A_3, S_0>$		(3a)
$<A_1, S_2, A_3, S_0, A_4>$	a	(2a)
$<A_1, S_2, A_3, B_0>$		(3b)
$<A_1, S_2, A_3>$	b	(2b)
$<A_1, S_2, A_5, S_1>$		(3a)
$<A_1, S_2, A_5, S_1, A_6>$	a	(2a)
$<A_1, S_2, A_5, B_1>$		(3b)
$<A_1, S_2, A_5>$	b	(2b)
$<A_1, B_2>$		(3b)
$<A_1>$	b	(2b)
$<S_3, B_4>$		(3a)
$<S_3, B_4, A_7>$	a	(2a)
$<S_3, B_4, S_4, B_3>$		(3a)
$<S_3, B_4, S_4>$	b	(2a)
$<S_3, B_5>$		(3b)
$<S_3>$	b	(2b)
		(3c)

定理1．Φ(G, D)が生成する文字列はL(G)の文であり、真の入れ子の深さが2Dを越えない構文木を持つ。また、真の入れ子の深さがD－1を越えない構文木を持つ文はΦ(G, D)によって生成される。

実際の運用に現われる真の入れ子の深さの最大値に1を加えた値がDだと考えれば、Φ(G, D)は人間の発話できる正しい文を全て生成することができる、ということが、上の定理から言える。

しかしながら、話はむしろ逆で、Φ(G, D)として抽象化されるような言語処理過程の結果として、真の入れ子に関する制約があるのだ、と考えるべきである。われわれはΦ(G, D)のsを人間の短期記憶ないしはその一部の抽象化と考えているので、上の手続きの各ステップ中のdを、その瞬間の記憶負荷と呼ぼう。ある文の記憶負荷と言った場合には、その文をΦ(G, D)が生成するときのdの最大値を指すものとする。もちろんその場合、その文のひとつの構文木が想定されているものとする。Φ(G, D)を発話のモデルと考えるからには、短期記憶の有限性によって制約を受けるのは真の入れ子の深さではなく、記憶負荷であると考えなければならない。

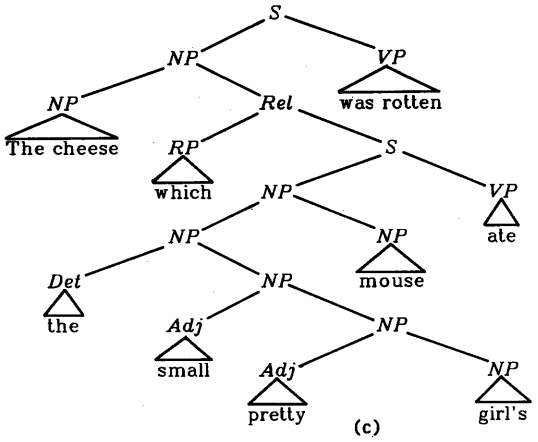
このことは次のふたつの例文が何とか発話できるものであることによって例証される。

(c) "The cheese which the small pretty girl's mouse ate was rotten."

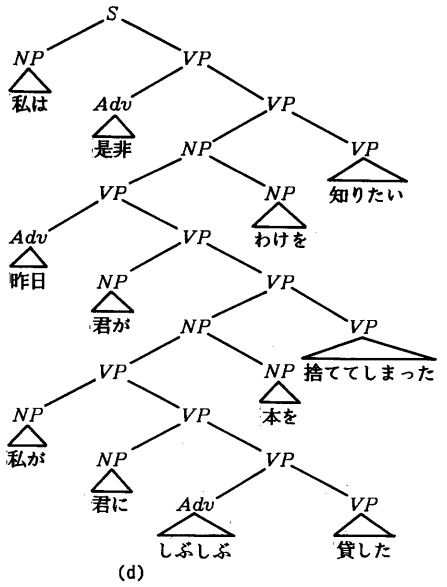
(d)「私は是非昨日君が私が君にしぶしぶ貸した本を捨ててしまったわけを知りたい。」

図4にこれらの構文木を示す。

(c), (d)の真の入れ子の深さはそれぞれ4、5であり、(a), (b)と一致している。ところが、(a), (b)の記憶負荷がそれぞれ5、6であるのに対し、(c), (d)ではそれぞれ2、3である。つまり、発話の難しさを引き起こしているのは、真の入れ子の深さというよりはむしろ記憶



(c)



(d)

図4

負荷なのである。

そういう文脈の中でΦ(G, D)を特徴づけるのが次の定理である。実際には、定理1はこれから導かれる。

定理2．Φ(G, D)がある文字列を生成できる条件は、その文字列がGで生成される文であり、かつその記憶負荷がある構文木に関してDを越えないことである。

6．実時間の構文解析

この節では、上のΦ(G, D)をもとにして、自然言語の構文解析を実時間で行うアルゴリズムを設計する。

前に示した例文を見ると、認識の難しさについても真の入れ子の深さよりはむしろ記憶負荷が大きな原因になっているように思われる。そこでわれわれは、ここで設計するアルゴリズムが満足すべき要件として、次のふたつを考える。

(ア) そのアルゴリズムがある文字列αを受理するならば、αはL(G)の文であり、このとき記憶負荷がDを越えないようなαの構文木がつくられる。

(イ) L(G)の文αが記憶負荷がDを越えない構文木を持てば、そのアルゴリズムはαを受理する。

Φ(G, D)の入出力関係を入れかえ、さらに構文木をつくる手続きを加えれば、定理2によって、上の(ア)、(イ)を満足するアルゴリズムができるだろう。

以下で述べるアルゴリズムにおいて構文木を作る際に、未完成の木の部分構造は、図2のR1, …, Rd のような構造のリストで表わすことにする。一般に、構文木の部分木であって、右端の葉が非終端記号で、他には欠けたところのないものを左木と呼ぶ。左木の右端の葉をその左木の尾と呼ぶ。構文木の集合を TREE，左木の集合を LTREE と書くことにする。図2のRiはAi が根でありBiが尾であるような左木である。また、

$LTLIST = \bigcup_{d=0}^{D} LTREE^d$, $TLIST = LTLIST \times TREE$

とする。

アルゴリズム1．

2つの変数 LTL⊂LTLIST, TL⊂TLIST を用いる。

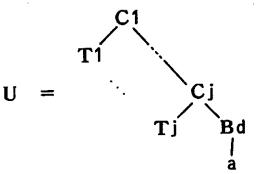
(1) LTL：＝{＜ ＞} として(2)へ。

(2) まず TL：＝φとし、M∈LTLなる全てのMに対して、次の(2a), (2b)を実行して(3)へ。ただし、M＝＜R1, …, Rd＞であるとし、次の入力は a ∈ VTであるとする。

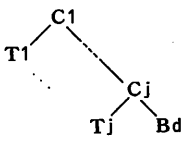
(2a) A→aなる全てのA ∈ VNに対し、TL：＝TL ∪ {＜R1, …, Rd, $\overset{A}{\underset{a}{|}}$ ＞} とする。

(2b) d⩾1かつBd→aならば TL：＝TL ∪ {N} とする。ただし、N＝＜R1, …, Rd-1, U＞

U ＝



Rd＝



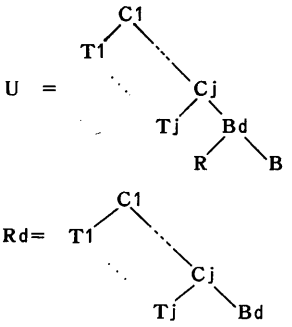
である。

(3) 入力が残っているとき：

まずLTL：＝φとし、M∈TL なる全てのMに対して、次の(3a)，(3b)を実行して(2)へ。ただし、M＝＜R1，…，Rd，R＞とし、木Rの根をC∈VNとする。

(3a) d＜Dならば、A→CBなる全てのA，B∈VNに対して、LTL：=LTL U {＜R1，…，Rd，$\overset{A}{\underset{R\ \ B}{/\backslash}}$ ｜＞とする。

(3b) d≧1ならば、Bd →CBなる全てのB∈VNに対し、LTL：=LTL U {N} とする。ただし、N＝＜R1,…, Rd-1, U＞



である。

入力が残っていないとき：

＜T＞∈TLかつ根がSであるようなTを全て出力して終わる。

定理3．アルゴリズム1は(ア)、(イ)を満足する。

ところが、上のアルゴリズムにおいて、LTL と TL の濃度は入力の長さに関して指数関数的に増大するので、その時間計算量は指数関数になる。そこで、(ア)、(イ)に違反しない範囲でLTLの要素を削除することを考える。左木Rの根をRM(R)，尾をTM(R)と書き、

σ（＜R1，…，Rd＞）

＝＜RM(R1)，TM(R1)，…，RM(Rd)，TM(Rd)＞

としよう。

アルゴリズム2．

このアルゴリズムは上のアルゴリズム1の(3) の後に(4)を実行するようにしたものである。

(4) M，N∈LTL，M≠Nなる全てのM，Nに対し、σ(M)＝σ(N)ならば、LTL：=LTL－｛M｝として(2)へ行く。

定理4．アルゴリズム2は(ア)、(イ)を満たし、入力の長さに関し線型の時間計算量を持つ。

アルゴリズム2は(4)において、Φ(G，D）の同じ状態に対応する LTLの異なる要素のひとつを捨てることによって、LTL の濃度を$\sum_{d=0}^{D}|VN|^{2d}$ で抑えているわけである。しかし、|VN| は少くとも10程度、Dは3ぐらいだろううから、この上限はかなり大きなものになってしまう。実時間の処理を実現するためには、アルゴリズムが入力を順方向に処理しその時間計算量が入力の長さに関して線型であれば数学的には十分であるが、現実問題としては、時間計算量の比例定数が十分小さくなければならない。

そこで、上のアルゴリズムの(4)を改良してLTLの濃度をさらに小さく保つことを考える。

Ψ＝(Q，Σ，q_I，F，δ）を非決定性有限オートマトンとしよう。Qは状態の集合、Σはアルファベットの集合、q_I ∈ Qは初期状態、F ⊆ Qは最終状態の集合、δ：Q×Σ*→2^Qは遷移関数である。このようなΨに対し、

関数SUF_Ψ：Q →2^{Σ^*} を

$$SUF_\Psi(q)=\{\alpha\in\Sigma^*\mid\delta(q,\alpha)\cap F\neq\phi\}$$

で定めよう。さらに、Qにおける半順序 $\overset{\Psi}{\lhd}$ を

$$q\overset{\Psi}{\lhd}r\ \longleftrightarrow\ SUF_\Psi(q)\subseteq SUF_\Psi(r)$$

によって定める。

定理5．$q\overset{\Psi}{\lhd}r$のとき、しかもそのときに限り、

$|\alpha|<2^{2|Q|}$なる全ての$\alpha\in\Sigma^*$ に対して、

δ(q，α)∩F≠φならばδ(r，α)∩F≠φである。

この定理によって、$\overset{\Psi}{\lhd}$の構造を陽に求めるアルゴリズムが存在することがわかる。そこで、$\overset{\Phi(G,D)}{\lhd}$を単に ◁ と書くことにすると、次のアルゴリズム3は確かにアルゴリズムになっている。

アルゴリズム3．

このアルゴリズムは(4) を除いてアルゴリズム2と同じである。

(4)M，N∈LTL，M≠Nなる全てのM，Nに対し、σ(M) ◁ σ(N) ならば、LTL：=LTL－{M} として(2)へ。

定理6．アルゴリズム3も(ア)、(イ)を満たし、入力の長さに関して線型の時間計算量を持つ。

◁ の構造はアルゴリズム3の実行に先立って求めて

おけばよいから、アルゴリズム3の効率は $\lhd$ の構造を求める計算の複雑さには影響されない。

ところが、定理5を見ると、$\lhd$ の構造を求めるには少くとも $0(2^{2^{|Q|}})$ 程度の手間がかかりそうである。この場合は $|Q|>|VN|^{20}$ であるから、$|VN|=10, D=3$ としても、この計算はとても手に負えない。

そういうわけで、$\lhd$ を正確に計算することは諦めざるを得ない。そのかわり、アルゴリズム3の(4)において、何らかの方法で $\sigma(M) \lhd \sigma(N)$ がわかったものについてだけ LTL:=LTL-{M} とすることにしよう。

一般に、次の定理に示されるように、$\lhd$ を隣接する非終端記号の組ごとの関係に分解することができる。

定理7．次の条件(A),(B),(C)が成り立つとき、

$\langle A1, B1, \cdots, Ad, Bd\rangle \lhd$

$\langle C1, D1, \cdots, Cd, Dd\rangle$ である。

ただし、wider-than(B, A, D, C) という述語は、

$E0=B, En=A$ で、$0 \leqslant j < n$ に対して

$Ej \rightarrow Ej+1Fj$ であるような $n \geqslant 0$ と

$E0, \cdots, En, F0, \cdots, Fn-1 \in VN$ が存在するならば、

$G0=D, Gn=C$ で、$0 \leqslant j < n$ に対して

$Gj=Gj+1Fj$ であるような $G, \cdots, Gn \in VN$ が存在する。

という意味であるとする。

(A) wider-than(S, A1, S, C1)。

(B) $1 \leqslant i < d$ に対して、

wider-than(Bi, Ai+1, Di, Ci+1)。

(C) 任意の $a \in VT$, $A \rightarrow a$ なる任意の $A \in VN$ に対し、wider-than(Bd, A, Dd, C) かつ $C \rightarrow a$ なる $C \in VN$ が存在する。

wider-than(B, A, D, C)の値を全てのB, A, D, Cの組み合わせに対してあらかじめ求めておいて、上の定理を使って $\sigma(M) \lhd \sigma(N)$ がわかるかどうかを調べるというやりかたは、$\lhd$ を正確に求めるよりは、はるかに現実的である。しかし、実際の自然言語に見られる構文的な特徴を考慮することによって、さらに効率のよい方法が得られる。

たとえば、文法Gが次の性質を満たすと仮定しよう。

(LR) 全てのA, B, C, E, Fに対して、もし $A \rightarrow FB$ ならば、$C \rightarrow AE$ のとき、しかもそのときに限り $C \rightarrow BE$ である。

これを直観的な言い方に直すと：

句全体の文法的な性質は、その句の右端の部分によって決定される。

日本語、朝鮮語、トルコ語など、修飾語が被修飾語に先立つような諸言語は、おおよそこの性質を満たしている。こうした言語は世界中の言語の半数近くを占めている[10]ので、(LR)を前提としたやり方は広い適用可能性をもっていると考えられる。

定理8．Gが(LR)を満たすとき、次の(A),(B) が成り立てば、

$\langle A1, B1, \cdots, Aj, Bj\rangle \lhd$

$\langle C1, D1, \cdots, Ck, Dk\rangle$ である。

ただし、A, B $\in$ VNに対して、

$\{\alpha \in VT \cup VN^2 \mid A \rightarrow \alpha\} \subseteq \{\alpha \in VT \cup VN^2 \mid B \rightarrow \alpha\}$

を $A \leqslant B$ と書く。

(A) $A1=S$ ならば $C1=S$ である。

(B) 自然数から自然数への関数fで、

$1 \leqslant f(1) < f(2) < \cdots < f(k) \leqslant j$ かつ

$1 \leqslant i \leqslant k$ に対して $Bf(i) \leqslant Di$ となるものが存在する。

日本語や朝鮮語とは逆に、修飾語が被修飾語より後に置かれる言語に関しても、左右を逆転した議論によって、やはり上の定理と同程度に便利な結果を得ることができる。修飾の向きに左右両方がある英語などの言語については、これらの結果を混用することによって対処できる。

7．実験

前節で述べたアルゴリズムが本当に実時間で動くことは、実験による検証を必要とする。そこでわれわれは、日本語を対象に、このアルゴリズムに基づいた構文解析システムを作成し、簡単な実験を行なった。日本語は最も馴染みの深い言語であるのみならず、性質(LR)をほぼ完全に満足するという、比較的すっきりした構造をもっているので、最初の実験の対象として適切であると考えたわけである。

われわれが用いた文法は各文節を終端記号とする文脈自由文法である。この文法は非常に簡単なもので、陳述副詞と述語の対応、並列句の扱いなどの細かい記述は含んでいないが、助詞と用言の係り受けなどの基本的なものはほぼ盛り込んである。

実際に時間がかかるのは、構文木を作る手続きよりも文節を認識する手続きであるが、それを含めてもわれわ

れの実験システムは十分速いことがわかった。解析された文は文節の数が15個程度までのものであるが、LTL の濃度は文節の数が5,6 個を過ぎたあたりから安定しており、この傾向はいかに長大な文であろうとも変わらないと考えられる。したがって、このシステムは実時間の構文解析をしていると見なすことができる。

8．認識のモデルに向けて

(LR)が満たされているとき、定理8からも想像されるように、図2の斜線部分と白い部分の整合性を管理するために、R1, …, Rdの尾だけを短期記憶に入れておけばよい。つまり、統語構造が左から右への修飾関係のみによって構成されている言語では、構文木を生成しているときに短期記憶に入っている非終端記号の数は、記憶負荷に等しい。

統語構造が右から左への修飾関係だけで構成されるような言語に関しても、同様の議論が成立する。すなわち、記憶負荷と同じ個数の非終端記号で生成の際の構文的な整合性が管理できる。

したがって、人間の短期記憶における1個の情報の塊がちょうど1個の非終端記号に対応すると仮定すれば、少くとも左から右への修飾だけで文ができる言語、および右から左への修飾だけで文ができる言語に関しては、統語的な制御に用いられる情報の塊の数は記憶負荷程度だということになる。

記憶負荷の値は文法によって変わるが、たとえば日本語の場合、ごく普通に作られた文法に関して、生成に際しても認識に際しても、記憶負荷の最大値は高々4程度と思われる。したがって、短期記憶中に3±2個の情報の塊を別に確保することができ、これらは意味的な処理や構文的な曖昧さを解決するための処理に使われているのであろう。

ところが、6節で述べたわれわれのアルゴリズムは、以上の議論にあてはまらない。つまり、LTL の元の中に含まれる異なる左木の個数が短期記憶中の情報の塊の数に一致すべきであるにもかかわらず、その数が陽に7±2で抑えられていない。実験においても、その数は最大10以上に達していることがわかった。

これはわれわれのアルゴリズムが条件(ア), (イ)を満足するために構文的な曖昧性の解決を完全に行なっているからである。ところが人間が文を認識する際には、少くとも条件(イ)は満たされていない。Marcus[11]はここに着目し、garden path を排除することによって、人間が普通に行なう認識手順を決定性の手続きで模倣することを試みている。われわれの方法は garden path を排除しておらず、Marcusの方法は真の入れ子の深さの有限性に合致していないので、両者は相補う関係にあり、これらを結びつけることによって、言語の認識過程のモデルができるのではないかとわれわれは考えている。

[参考文献]

[1] Woods, W.A., "Transition Network Grammars for Natural Language Analysis", *CACM*, Vol. 13, (October 1970), pp. 591-606.

[2] Pereira, F.C.N., and D.H.D. Warren, "Definite Clause Grammar for Language Analysis – A Survey and a Comparison with Augmented Transition Networks", *Artificial Intelligence*, Vol. 13, (1980), pp. 231-278.

[3] Pratt, V.R. "LINGOL – A Progress Report" *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence*, Vol. 1, (1975), pp. 422-428.

[4] Miller, G.A., "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", *The Psychological Review*, Vol. 63, No. 2 (March 1956), pp. 81-97.

[5] Klatzky, R.L., *Human Memory: Structures and Processes (2nd edition)*, (1975), W. H. Freeman and Company.

邦訳：「記憶のしくみⅠ, Ⅱ」
箱田；中溝訳　心理学叢書, サイエンス社

[6] Gazdar, G., "Phrase Structure Grammar", in P. Jacobson and G. K. Pullum (*eds.*) *On the Nature of Syntactic Representation*, (1981), Reidel: Dordrecht.

[7] Gunji, T., *A Phrase structural Analysis of the Japanese Language*, M.A. thesis, Ohio State University, (1981).

[8] Wetherell, C.S., "Probabilistic Languages; A Review and Some Open Questions", *Computing Surveys*, Vol. 12, No. 4 (December 1980), pp. 361-379.

[9] Hasida, K., *A Real-Time Syntactic Analysis of Natural Languages*, M.S. Thesis, University of Tōkyō, (January 1983).

[10] Greenberg, J.H., "Some Universals of Grammar with Particular Reference to the Order of Meaningful Elements", in Greenberg (*ed.*) *Universals of Grammar*, (1963), pp. 73-113.

[11] Marcus, M.P., *A Theory of Syntactic Recognition for Natural Language*, (1980), MIT Press: Cambridge, Mass.