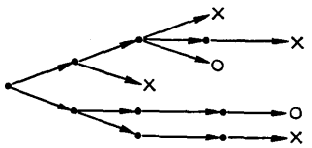
Context-Free Phrase Structure Language における subphrase の analyzability と一般構造分析理論*

金　山　　裕**

1. はじめに

筆者は先に context-free PSL における基本構造分析法を発表した[1]．この分析法の特長は，ある文法が与えられたとき，その文法に対する知識を何も持っていなくても，各 phrase を subphrase に分割する仕事さえすれば，如何なる string をも分析してしまうことである．しかし，この方法は結局のところ，可能なあらゆる推論をしらみつぶしに調べてゆく方法であるから，無駄な手順を踏む可能性がある．

言語における構造分析の手順は，自然語，形式語のいかんを問わず，推論の岐路があり，その岐路が先へ行くに従いふえてゆく傾向がある．これは数学の各公論系における定理の証明，ゲームを解く問題，などの自動化に見られる共通の特徴である．context-free PSL においてはその岐路がどれほどふえても，必らず有限手順で終ることは，基本構造分析理論において述べた．しかしこの種の問題においては，無効な推論をなるべく早目に切り捨て，有効な推論のみをとりあげてゆくことが大切である（**第 1.1 図**参照）．



第 1.1 図　構造分析における推論の岐路

基本構造分析法においては，各状態で direct analyzable な subphrase unit はすべて分析に採用していた．しかし本論文では，analyzable というさらに厳しい条件に合格した subphrase unit のみを採用する．これによって，従来の delimiter 間の優先順位という概念のもつ意味がはっきりした．そして以下に示す一般構造分析法より能率の良い分析は（文法の形を変えないことには）不可能であることも示した．

以下の記述において未定義の術語は前論文の定義にしたがう[1]．

2. analyzable な subphrase unit

分析をなるべく速く行なうには，

（1） step 数をなるべく少なくすること，

（2） 分析岐路をなるべく少なくすること，

の二とおりの手段がある．しかし，string が与えられたときそれを分析するに要する step 数は，分析過程に出てくる subphrase の数に等しく，さらにその subphrase は文法によってきまってしまう．したがって，文法を変更しない限り step 数を変えることはできない．

一方 (2) についてみると，基本構造分析法での欠点は前論文で指摘したとおりである[1]．その行き方を基にたとえるなら，「いよいよ打つ手が無くなってはじめて失敗したことを悟る」ようなものであった．簡単な文法を例にとる．

例 2.1

$$\left.\begin{aligned} &G=\{R_1, R_2, R_3, R_4, R_5\} \\ &R_1: S \rightarrow S+A=S+\circ A \\ &R_2: S \rightarrow A \\ &R_3: A \rightarrow A\times B=A\times \circ B \\ &R_4: A \rightarrow B \\ &R_5: B \rightarrow V \end{aligned}\right\} \quad (2.1)$$

なる文法を与えると，subphrase unit は 7 個となる．

$$\left.\begin{aligned} &\alpha_1=[S+, 1, 1, \#, 0] \\ &\alpha_2=[A, 1, 2, S, -] \\ &\alpha_3=[A, 2, 1, S, 0] \\ &\alpha_4=[A\times, 3, 1, \#, 0] \\ &\alpha_5=[B, 3, 2, A, -] \\ &\alpha_6=[B, 4, 1, A, 0] \\ &\alpha_7=[V, 5, 1, B, 1] \end{aligned}\right\} \quad (2.2)$$

この文法は V（変数を想定している）を 1 個以上，

---

* Analyzability of Subphrases and General Theory of Syntax Analysis in Context-Free Phrase Structure Language, by Yutaka Kanayama (Graduate School of Tokyo University, now with the Central Laboratory of Hitachi Ltd.)

** 東京大学大学院数物系研究科，現在日立製作所中央研究所．

＋または×によってつなげた sentence を生成する．$V+V\times V$ なる sentence の基本構造分析法による分析を**第2.1図**に示す．このような岐路ができる理由を考えてみる．

定義 2.1

基本構造分析法のある段階におけるシステムの状態を表わす情報の一部．

$$\gamma=\gamma(W;\ K,\ L) \tag{2.3}$$

は，次の意味を有する．

（1） residual string, q について，

$$W_h\backslash q \tag{2.4}$$

が成立するすべての W_h のうち最長のものを W で指定する．(2.4) 式を満足する subphrase が存在しないときは，

$$W=\Lambda \tag{2.5}$$

とする（この場合には direct analyzable な subphrase unit は存在せず，trivial のようにみえるが，やはり考慮の対象としなければならない）．

（2） partial stack, S_p について，それが空であるとき，すなわち

$$j=0 \tag{2.6}$$

であるとき，

$$K=L=0 \tag{2.7}$$

とする．一方 S_p が空でないとき，すなわち

$$j\geqq 1 \tag{2.8}$$

なるとき

$$\left.\begin{aligned}K&=KP_j\\ L&=LP_j\end{aligned}\right\} \tag{2.9}$$

とする． （定義 2.1 終り）

分析の途中の状態 γ は，q と S_p の内容によって，具体的に W, K, L を用いて表わされる．一般に G が与えられると，相異なる subphrase unit および Λ が W としてとり得る string である．また，tail でない subphrase の数に1を加えただけ（K, L）の組合わせがある（S_p が空の場合があるため）．

例 2.2

例 2.1 の G において，相異なる subphrase は

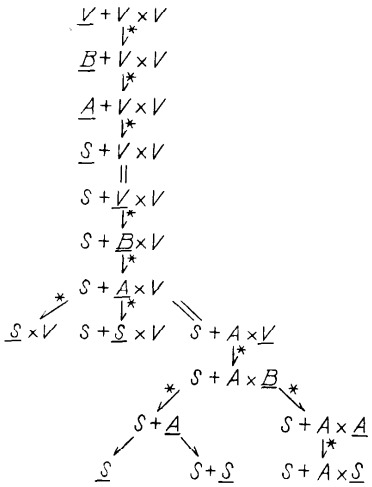
$$S+,\ A\times,\ A,\ B,\ V \tag{2.10}$$

の5個である．また，tail でない（$X_h=\#$ なる）subphrase unit は α_1, α_4 の二つである．つまり partial stack に関係する可能性のある α_h はこの二つだけである．これからすべての可能な γ をマトリクスの形で示すことができる．この各状態について，direct analyzable な subphrase unit を対応させたのが**第2.1表**である．$W=\Lambda$ なるときは，もちろん，direct analyzable な α_h は存在しない．

第2.1表 各状態において direct analyzable な subphrase unit

W		$S+$	$A\times$	A	B	V	Λ
K	L						
0	0	α_1	$\alpha_3,\ \alpha_4$	α_3	α_6	α_7	—
1	1	α_1	$\alpha_2,\ \alpha_3,\ \alpha_4$	$\alpha_2,\ \alpha_3$	α_6	α_7	—
3	1	α_1	$\alpha_3,\ \alpha_4$	α_3	$\alpha_5,\ \alpha_6$	α_7	—

第 2.1 図において，分析の終止状態は5個あるが，成功しているのは1個所である．それ以外の状態は $W_h\backslash q$ なる W_h が存在しない（$W=\Lambda$）ために分析できない．



第 2.1 図 基本構造分析法による分析例

分岐点は3個所ある．上から 1, 2, 3 と番号をつける．それらは，第 2.1 表において，

$$W=A\times;\ K=1,\ L=1 \tag{2.11}$$

$$W=B;\ K=3,\ L=1 \tag{2.12}$$

$$W=A;\ K=1,\ L=1 \tag{2.13}$$

なる状態に相当する．それらの状態において direct analyzable な subphrase unit はそれぞれ3個，2個，2個であるから，岐路もそれに等しいだけある．（例 2.2 終り）

直観的にみると例 2.1 の G は素直な文法であり，もちろんあいまいではない．したがって基本構造分析法を改善する余地があるのではないか，ということが考えられる．

構造分析の目的は tree structure を求めることで

あった．したがって，分析の各段階で使う direct analyzable な α_h が見込みのあるものであるかどうかは，あらかじめできる限り調べておくことが望ましい．分析に成功する可能性が全くない α_h を考慮の対象から除くことは，詰碁において，合法的ではあるが見込みの全くない手を捨てることに相当する．推論をたてる上で，この種の「読み」は最低限必要であろう．

「成功の可能性が全然なくはない」という概念を次のように定義する．

定義 2.2

γ において，direct analyzable な subphrase unit が次の条件を満たすとき，analyzable であると定義する．

（条件）

システムの状態が γ であるような少なくとも一つの main string，s_1 について，α_h を用いた analysis，

$$s_1 \overset{*}{\rightarrow} s_2 \quad \text{または} \quad s_1=s_2 \tag{2.14}$$

を実行したのち，

$$s_2 \overset{*}{\Rightarrow} S \tag{2.15}$$

なる analysis が少なくも一つ存在する．

例 2.3

例 2.1 の G において，analyzable な α_h を示したのが**第 2.2 表**である．direct analyzable な α_h のうちのあるものは analyzable ではない．

第 2.2 表　各状態において analyzable な subphrase unit

W		$S+$	$A\times$	A	B	V	Λ
K	L						
0	0	α_1	α_4	α_3	α_6	α_7	—
1	1	—	α_4	α_2	α_6	α_7	—
3	1	—	—	—	α_5	α_7	—

たとえば，$\gamma(A\times;\ 1,\ 1)$ なる状態（第 2 行，第 2 列）において，α_2 は direct analyzable であるが，analyzable ではない．その理由を次に示す．

この状態では，α_1 が partial stack に入っており，

$$A\times\backslash q \tag{2.16}$$

であるから，main string は

$$\begin{aligned} s_1&=pq \\ &=p_1S+\underline{A}\times q_1 \end{aligned} \tag{2.17}$$

ここで下線は $c(q)$ を表わし，

$$p=p_1S+,\quad q=\mathrm{A}\times q_1 \tag{2.18}$$

ここで α_2 を使って analysis をすれば，

$$s_1=p_1S+\underline{A}\times q_1 \overset{*}{\rightarrow} p_1S\times q_1 \tag{2.19}$$

ところで，いかなる p_1，q_1 についても

$$S \Rightarrow p_1S\times q_1 \tag{2.20}$$

は成立しない．つまり $S\times$ なる string は生成されない．よって α_2 は $\gamma(A\times;\ 1,\ 1)$ において analyzable でない（例 2.3 終り）．

定理 2.1

構造分析においては，各状態において analyzable でない subphrase unit を使って分析する必要はない．

（証明）

analyzable でない subphrase unit は，たとえ direct analysis は可能であっても，いつかは失敗する．　Q.E.D.

この性質があるので，基本構造分析法よりずっと能率的な分析が可能となる．ところで，任意の γ において，任意の α_h の analyzability は決定可能であろうか，という疑問が生ずる．

定理 2.2

状態 γ において，α_h が analyzable であるか否かは決定可能である．

（証明）

システムの状態を

$$\gamma=\gamma(W;\ K,\ L) \tag{2.21}$$

とする．また問題とする subphrase unit は

$$\alpha_h=[W_h,\ K_h,\ L_h,\ X_h,\ B_h] \tag{2.22}$$

とする．まず，α_h が direct analyzable でなければ analyzable でない．これは決定可能である．α_h が direct analyzable であれば定義によって，$W_h\backslash W$ である故，

$$w=W_h\backslash W=\text{constant} \tag{2.23}$$

とおく．main string s_1 は一般に partial string と residual string の積であり，

$$s_1=pq \tag{2.24}$$

(2.4) 式より，$q_1=W\backslash q$ とおくと

$$\begin{aligned} s_1&=pq \\ &=pWq_1 \\ &=pW_hwq_1 \end{aligned} \tag{2.25}$$

（1） $K\neq 0\neq L$ なるとき，

partial stack の最上段の内容は partial string の一部（または全部）を構成し，

$$\begin{aligned} P&=p_1x_{K1}\cdots\cdots x_{KL} \\ &=p_1y_{KL} \end{aligned} \tag{2.26}$$

(1.1) α_h が head でなく，tail であるとき，

W_h は y_{KL} につながって phrase を完成する.

$$\begin{aligned} s_1 &= p_1 y_{KL} W_h w q_1 \\ &= p_1 x_h w q_1 \\ &\overset{*}{\rightarrow} p_1 X_h w q_1 \end{aligned} \tag{2.27}$$

したがって, α_h が analyzable であるためには, 与えられた $X_h w$ と任意の p_1, q_1 について

$$p_1 X_h w q_1 \overset{*}{\Rightarrow} S$$

すなわち (2.28)

$$S \Rightarrow p_1 X_h w q_1 \tag{2.29}$$

なる analysis が少なくとも一つ存在するか否かが問題となる. ところでこれが決定可能であることはすでに Bar-Hillel らが証明している[2](注). よって α_h が analyzable であるか否かは, このばあい決定可能である.

1.2) α_h が head でなく, tail でないとき

W_h は y_{KL} につながるが phrase は完成しない. α_h が analyzable であるためには

$$\begin{aligned} s_1 &= p_1 y_{KL} W_h w q_1 \\ &= p_1 y_{K,L+1} w q_1 \\ &= s_2 \\ &= p_1 y_{K,L+1} q_2 q_3 \\ &\overset{*}{\rightarrow} p_1 X_K q_3 \\ &\overset{*}{\Rightarrow} S \end{aligned} \tag{2.30}$$

ただし

$$w q_1 = q_2 q_3 \tag{2.31}$$

かつ

$$y_{K,L+1} q_2 \overset{*}{\rightarrow} X_K \tag{2.32}$$

いいかえると, 与えられた $y_{K,L+1}$, w, X_K と任意の p_1, q_2, q_3 について,

$$\left.\begin{aligned} &S \Rightarrow p_1 X_K q_3 \\ &X_K \rightarrow y_{K,L+1} q_2 \\ &w \backslash q_2 q_3 \end{aligned}\right\} \tag{2.33}$$

を同時に満足することがあるか否かが問題となる. これが上の 1.1) と異るのは, w なる string が $q_2 q_3$ の left substring として含まれなければならない. という条件だけである. w の長さが有限であるからこの検査を含めても, 1.1) と同じ理由により決定可能である.

1.3) α_h が head であり, tail であるとき

α_h は y_{KL} につながらず, それだけで phrase をなす. α_h が analyzable であるためには

$$\begin{aligned} s_1 &= p_1 y_{KL} W_h w q_1 \\ &\overset{*}{\rightarrow} p_1 y_{KL} X_h w q_1 \\ &= s_2 \\ &= p_1 y_{KL} q_2 q_3 \\ &\overset{*}{\rightarrow} p_1 X_K q_3 \\ &\overset{*}{\Rightarrow} S \end{aligned} \tag{2.34}$$

が成立しなければならない. ただし

$$\left.\begin{aligned} &X_h w q_1 = q_2 q_3 \\ &X_K \rightarrow y_{KL} q_2 \end{aligned}\right\} \tag{2.35}$$

これは 1.2) と同様にして決定可能である.

1.4) α_h が head であり, tail でないとき,

α_h は y_{KL} につながらず, 完全な phrase ともならない. α_h が analyzable であるためには,

$$\begin{aligned} s_1 &= p_1 y_{KL} W_h w q_1 = s_2 \\ &= p_1 y_{KL} W_h q_2 q_3 \\ &\overset{*}{\rightarrow} p_1 y_{KL} X_{Kh} q_3 \\ &= p_1 y_{KL} X_{Kh} q_4 q_5 \\ &\overset{*}{\rightarrow} p_1 X_K q_5 \\ &\overset{*}{\Rightarrow} S \end{aligned} \tag{2.36}$$

が成立しなければならない. ただし,

$$\left.\begin{aligned} &X_K \rightarrow y_{KL} X_{Kh} q_4 \\ &q_4 q_5 = q_3 \\ &X_{Kh} \rightarrow W_h q_2 \\ &w \backslash q_2 q_3 \end{aligned}\right\} \tag{2.37}$$

これが成立するか否かは 1.2) と同様にして決定可能である.

(2) $K=L=0$ なるとき

partial string は empty であり, α_h は head である.

$$s_1 = W_h w q_1 \tag{2.37}$$

2.1) α_h が tail であるとき,

α_h が analyzable であるためには

$$\begin{aligned} s_1 &= W_h w q_1 \\ &\overset{*}{\rightarrow} X_h w q_1 \\ &\overset{*}{\rightarrow} S \end{aligned} \tag{2.38}$$

が成立しなければならない. これは 1.1) と同様にして決定可能である.

2.2) α_h が tail でないとき,

α_h が analyzable であるためには

$$\begin{aligned} s_1 &= W_h w q_1 = s_2 \\ &= W_h q_2 q_3 \\ &\overset{*}{\rightarrow} X_{Kh} q_3 \\ &\overset{*}{\Rightarrow} S \end{aligned} \tag{2.39}$$

が成立しなければならない. ただし

(注) 156 頁の定理 5.1

$$\left.\begin{array}{l} X_{Kh} \rightarrow W_h q_2 \\ w \backslash q_2 q_3 \end{array}\right\} \quad (2.40)$$

これは 1.2) と同様にして決定可能である.

以上ですべての場合をつくし, それぞれ決定可能であることが証明された. (証明終り)

例 2.1 の G においては, 従来の delimiter pair の考え方によると×が＋より優先順位が高い. このことは第 2.1, 2.2 表に示すように, $\gamma(A\times;\ 1,\ 1)$ においては α_4 が analyzable であるのに対し, α_2 と α_3 は analyzable ではないことに相当する. したがって, analyzability は delimiter (または演算子) pair 間の優先順位の概念を含んでいる. しかし次の二つの理由により前者のほうがさらに広い概念である.

第1に, 前者は γ と α_h における絶対的な関係であり, ほかに direct analyzable な α_h が存在するか否かを問題としない. 詰碁の局面において, 着手の善悪はそれ自体として判断が可能であるのと似ている. このことは複数の着手の間での相対的な善悪の判断の可能性を否定していない. 後者は 2 characters の間の相対的な関係を示すに過ぎない.

第2に, 前者はシステムの状態, γ により規定されている. 実際, 同一の W_h が (異なる γ において) 異なる subphrase に解釈されることは文法によっては起りうることである. 一方, delimiter pair の方法では, character 自身が個有の意味をもっている文法しか扱わないゆえ, 限られた方法となっている.

定義 2.3

状態 γ において analyzable な α_h の数を

$$NS(\gamma) \quad (2.41)$$

とする.

定義 2.4

$$NS(\gamma) \geqq 2 \quad (2.42)$$

なる γ を branch state と称する.

定義 2.5

branch state をもたない G を deterministic と称する.

定理 2.2

あいまいな文法は branch state をもつ (証明略).

例 2.4

第 2.2 表のすべての γ において,

$$NS(\gamma)=0 \text{ or } 1 \quad (2.43)$$

よって例 2.1 の G は deterministic であり, あいまいではない.

定義 2.6

$$NS(\gamma) \geqq 1$$

なる γ において analyzable な α_h をそれぞれ,

$$\alpha_{hh}(\gamma,\ 1),\ \alpha_{hh}(\gamma,\ 2),\ \cdots\cdots,\ \alpha_{hh}(\gamma,\ NS(\gamma)) \quad (2.44)$$

とおく.

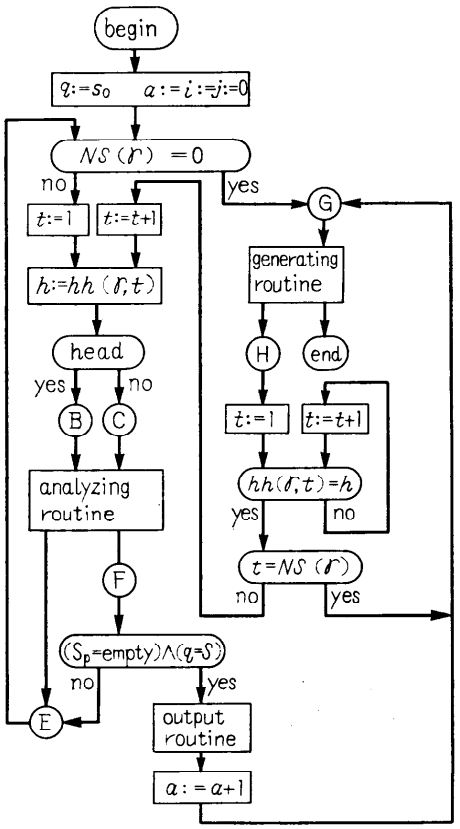
3. 一般構造分析法

本節で述べる分析法では, γ において analyzable な subphrase unit のみを分析に用いる. 基本構造分析法では direct analyzable なものを用いた点との違いである.

そのために, 定義 2.3 および 2.6 による $NS(\gamma)$ および $hh(\gamma,\ t)$ のリスト ($1 \leqq t \leqq NS(\gamma)$) を用いる. これはすべての γ について用意しておかなければならない. 基本構造分析法との異同は次のとおりである.

(1) analyzing routine は全く変らない.

(2) generating routine は, プログラムポイント H の前の, $h=f$ なる比較を除くだけであとは変らない.



第 3.1 図 一般構造分析法の流れ図

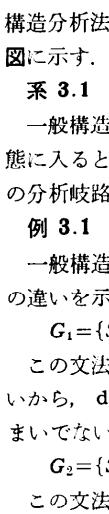
（3） decision routine は，direct analyzable であるか否かをみるのではなく，前記のリストによって直接に analyzable な α_h を求めるものになる．

（4） generating routine を出たあとは，(2.24) 式のリストを使い切ったか否かを調べて，analyzing routine に入るか generating routine に入るかをきめる．そのために t なるインデクスを使っている．

以上によって分析を行なうのが一般構造分析法であって，これを**第 3.1 図**に示した．大筋は基本構造分析法と変らない．

具体的な文法が与えられると，γ に関係したリストの作り方とひき方が実際問題としては重要となる．たとえば第 2.2 表で $V\backslash q$ でありさえすれば，S_p に関係なく α_7 を使える．しかしこれは一般的な問題ではないからここでは述べない．

第2.1図と同じ文法と basic string を与えて一般構造分析法により分析した例を**第3.2図**に示す．分岐は生じない．



第3.2図 一般構造分析法による分析例

系 3.1

一般構造分析法において，γ なる状態に入ると，そこから $NS(\gamma)$ とおりの分析岐路が生ずる．

例 3.1

一般構造分析法からみた3種の文法の違いを示す．

$$G_1=\{S\to SA,\ S\to A\} \quad (3.1)$$

この文法は branch state をもたないから，deterministic で，かつあいまいでない．

$$G_2=\{S\to ASA,\ S\to A\} \quad (3.2)$$

この文法では状態

$$\gamma=\gamma(A;\ 1,\ 1) \quad (3.3)$$

は branch state であり，この A が subphrase x_{11}, x_{21} のいずれであるかの区別がつかない．しかし G_2 はあいまいではない．したがって分析に岐路が生じても，成功するのはたかだか1個所である．

$$G_3=\{S\to SS,\ S\to A\} \quad (3.4)$$

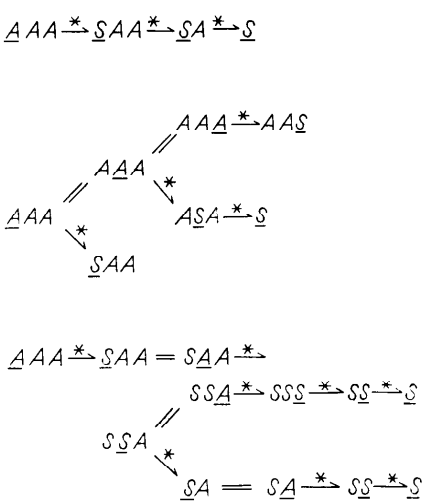
この文法はあいまいな文法であり，branch state,

$$\gamma=\gamma(S;\ 1,\ 1) \quad (3.5)$$

をもつ．

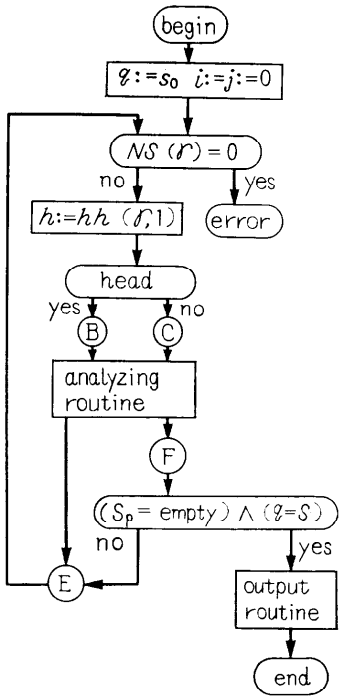
以上三つの文法はそれぞれ AAA なる sentence を有するが，その分析の手順を**第 3.3 図**に示す．

（例3.1終り）



第3.3図 3種類の文法における分析の手順

文法が deterministic であるときは，第3.2図(1)のように，岐路を生じないから，generating routine によって逆もどりする必要はなくなる．したがって，第 3.1 図よりずっと簡単な流れ図によって分析することが可能となる．それを**第 3.4 図**に示す．error と



第3.4図 deterministic な文法における一般構造分析法の流れ

記したのは，s_0 が sentence でないときの出口である．計算機用の入力言語のための構造分析はほとんどこれでよい筈である．

第 3.2 図に示した分析では，第 3.1 図の流れ図によっても，第 3.4 図の流れ図によっても実質的な差は生じない．しかし前者においては，分析に成功したのちも，branch state（この analysis には含まれていないが）をさがして，最初の状態へ逆戻りする．つまり，tree stack も partial stack も空になるまで generating routine を通って最初の状態へ帰る．一方，後者においては，いかなる状態も branch state でないことをあらかじめ知っているので，main string $=S$ となり，分析に成功すると，そこで止めてしまう．

4. 分析能率をさらに高める方法

一般構造分析法よりさらに能率的な分析法が存在するか否かを考えてみる．ここでも，与えられた文法をもとにして分析を行なう．という基本的な態度は相変らず守る．したがって，subphrase の集合は不変であるから，分析岐路を少なくすることが望まれる．

4.1 analyzability の概念

この概念を利用して分析に使う subphrase unit の候補者をしぼることによって，分析法はきわめて能率的なものとなった．さらに候補者をしぼることによって能率を高め得るか否かを検討する．

analyzability の決定にあたって用いたのは，partial stack の最上段と，residual string（の subphrase の長さまで）であった．これ以外の情報を知ることによって，さらに α_h の候補者をしぼり得るであろうか．

（1） tree stack の内容

これは分析の役に立たない．context-free PSG では，generation が string の他の部分と全く独立に行なわれるためである．

（2） residual string の内容

W_h を越えて右方を見ることは，subphrase または phrase を越えた分析を試みるのに等しい．これは文法の与え方を考えると無意味なことがわかる．

（3） $j\geqq 2$ なるときの partial stack の内容

上から2段目以下を調べて役に立つかといえば，これまた役に立たない．phrase が途中に1段入ってくることによって，情報のつながりが断ち切られてしまうためである．

以上によって，analyzable の範囲は定義 2.2 より狭くはできない．

4.2 branch state での決定を延期すること

文法によっては，branch state を持つけれど，特殊な手法を使うことによって，generating routine に入ることなく分析を進め得るものがある．

例 4.1

ALGOL 60 は次の2個の規則を含む[3]．

$$\left.\begin{array}{l}\langle\text{primary}\rangle \rightarrow (\langle\text{arithmetic expression}\rangle)\\ \langle\text{Boolean primary}\rangle \rightarrow (\langle\text{Boolean expression}\rangle)\end{array}\right\} \quad (4.1)$$

各 phrase は，左かっこと残りの部分の 2 個の subphrase に分れる．分析の途中で左かっこが出てくると，それは，いずれの規則によって生じたものか判定できない．したがって，（$\backslash q$ なる状態は branch state となる．しかし，左かっこがどの規則によって生じたかの決定は保留しておき，それにつづく subphrase が出てから決定するという便法がある．そのために，（に対しては，たとえば 100 なる K_h を与え，それにつづく2個の subphrase に対してはそれぞれ 1000−1，100−2 なるリストの番号を与えておく．そして partial stack に $KP_j=100$ および，$LP_j=1$（すなわち左かっこ）が入っているときは，$K_h=100-1$ または $100-2$ なる subphrase unit のいずれとも一致がとれたとして扱えばよい．

さらに一般的には，

$$\left.\begin{array}{l}W_1W_2W_3W_4\\ W_1W_2W_3W_5\end{array}\right\} \quad (4.2)$$

なる二つの phrase が G 中に含まれるとき（$W_1\sim W_5$ は subphrase とする，$W_4\neq W_5$），規則番号の最終決定は W_1，W_2，W_3 を調べた後，W_4 または W_5 を見てから行なう．また，

$$\left.\begin{array}{l}W_1W_2W_4\\ W_1W_2W_5\\ W_1W_3\end{array}\right\} \quad (4.3)$$

なる3個の phrase を含む G があれば，W_4，W_5 には二重のリスト番号がつく（$W_2\neq W_3$，$W_4\neq W_5$），$W_1\sim W_5$ の規則番号はそれぞれ K，$K-1$，$K-2$，$K-1-1$，$K-1-2$ となる．

上記のものと似ているが少し異なる文法の例として次を挙げよう．

例 4.2

G が次の2個の phrase を含むとする．

$$\left.\begin{array}{l} W_1W_2 \\ W_4W_1W_3 \end{array}\right\} \quad (4.4)$$

ここで $W_1 \sim W_4$ は subphrase, $W_2 \neq W_3$, $W_1 \neq W_4$ とする. $W_4W_1W_3$ の規則番号が K なるとき,

$$\gamma(W_1;\ K,\ 1) \quad (4.5)$$

なる状態が branch state となることがある. これは例 4.1 の場合と似ているが, 同じ手法では解決できない. このようなときには, W_1 のような特別な subphrase が来たときは, それを partial stack にのせると同時に特殊なマークをつけておく, そして次に W_2 が来るか W_3 が来るかによって, W_1 なる subphrase の去就を決定する. (例4.2終り)

以上に述べた例は, 比較的まれに起るから, 一般構造分析法にはとり入れなかったが, それと組合わせることは容易である. いずれも, 文法によっては, きわめて有効な手段であるが, 「左から逐次分析する」という原則をいくらか越えている. これらはもちろん analyzability の概念とは独立なものである.

5. おわりに

本論文の目的は, context-free phrase structure language における構造分析の能率を高めることであったが, それに関する問題点はほとんど解明できたと思う. 一般構造分析法の特長は, 「如何なる文法が与えられても, 文法の知識を生かした, 能率のよい分析が可能である」ことである. また, この方法以外に一般的な分析方法はほとんど考えられない. 詰碁などにおける「読み」は, analyzability を決定する, 定理 2.2 の証明手順に相当する. 原理的にこれ以上深く読み得ないことを 4.1 に示した. ただし, 決定をあとまで持ち越す手法が使えることがある.

この研究の将来の方向は, 一つは文法の理論的拡張である. type 1 以上の phrase structure language における一般構造分析はすでに成功した[4]. しかし詳しい内容については別の機会に御報告したい. もう一つは実際の応用であるが, 次のようなものが考えられる.

(1) コンパイラ作成の自動化

構造分析は翻訳の仕事の一部を占めるに過ぎない. しかし, 本論文で述べたような一般的な方法が, syntax directed compiler の自動化への第一歩であろう. 少なくも構造分析に関する限り, プログラマの熟練は要しなくなった.

(2) man-machine communication

通常のプログラム言語にしろコマンド言語にしろ, quick response が要求される場合には, (従来のコンパイラと異なった) 人間に近い言語の認識方法が望ましい. 一般構造分析法はその点でたいへん適していると思う.

またこれまで用いた計算機用言語はいずれも deterministic なものであった. しかし, それを自然語に近づけてゆくと, 必らずしも deterministic でなくなるであろう. あいまいになる可能性さえある. そのとき, この方法は威力を発揮する.

なお, 本論文では実際の計算機プログラムにおいて重要であると思われる各種の string (q, W_h, W), γ, NS, hh, α_h スタックなどに関係した処理を如何にして実現するかについては全く触れなかった. ここにもリスト処理の応用分野があると思われる.

最後に, 種々の面でお世話になった東京大学工学部元岡達助教授および本論文の骨子となる analyzability の考察にあたって適切な助言を賜った大学院学生今津敦志氏に深く謝意を表します.

参考文献

1) 金山: Context-Free Phrase Structure Language における基本構造分析理論, 情報処理, **7**, 1, 1966 年 1 月, pp. 13～25

2) Y. Bar-Hillel, M. Parles and E. Shamir: On Formal Properties of Simple Phrase Structure Grammars, Zeitschrift für Phonetic Sprachwissenschaft and Kommunikationsforschung, **14** (1961), pp. 143～172

3) P. Naur, ed.: Revised Report on th Algorithmic Language ALGOL 60, C. ACM, **6**, 1, pp, 1～17

4) 今津, 元岡, 金山: 一般的な Phrase Structure Language とその構造分析, 40 年情処大会, 17.

(昭和41年2月12日受付, 同5月1日再受付)