日本語埋め込み文の構文解析における諸問題

井佐原 均，田中 穂積
（電子技術総合研究所）

1．はじめに

筆者らは日本語からの意味抽出システムを実現するため，現在，日本語の大規模文法の作成に取りかかっている。

日本語の埋め込み文の構造については従来，文法学者により左方枝分かれ構造であると言われてきた。久野[1]によって示された例を図1に示す。この構造は，人間が話者の発話につれて日本語文の理解をふくらませてゆく過程にうまく対応付けられているため，従来の日本語文法においては左方枝分かれ構造が自然なものとして受け入れられている。しかしながら，計算機処理の立場から検討するとこのような構造の文法は解析上，自然なものと言えるであろうか。

図2は，図1に示されている2つの文法規則；

$$\begin{cases} S \rightarrow NP \cdot VP \\ NP \rightarrow S \cdot NP \end{cases}$$

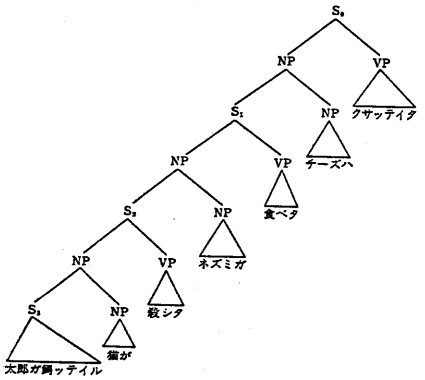
を用いて図1の例文を構文解析したものである。図2に示されるように，この文法規則は構文解析結果として複数の構文解析木を導出する。さらに複雑な文に対しては，導出される構文解析木の数は爆発的に増加する。図3に示すように，「花子は池に沈んだ人を助けた犬を見た。」という文に対し，実に12種類の構文解析木が導出されている。このような，構

ここで言う「左枝分れ」の意味は，次の例を見れば明らかになるのであろう。

(11) [[[太郎ガ飼ッテイル]S_3猫ガ殺シタ]S_2ネズミガ食ベタ]S_1チーズハ クサッテイタ。

(11)は，文法的な，しかも意味が通じる文である。この文は，次のような構造を持っているものと思われる。

(12)



上の構造図で，Sは文または節，[9] NP は名詞句 (Noun Phrase), VP

図1 文法学者による埋め込み文の解析
（参考文献1より P.5）

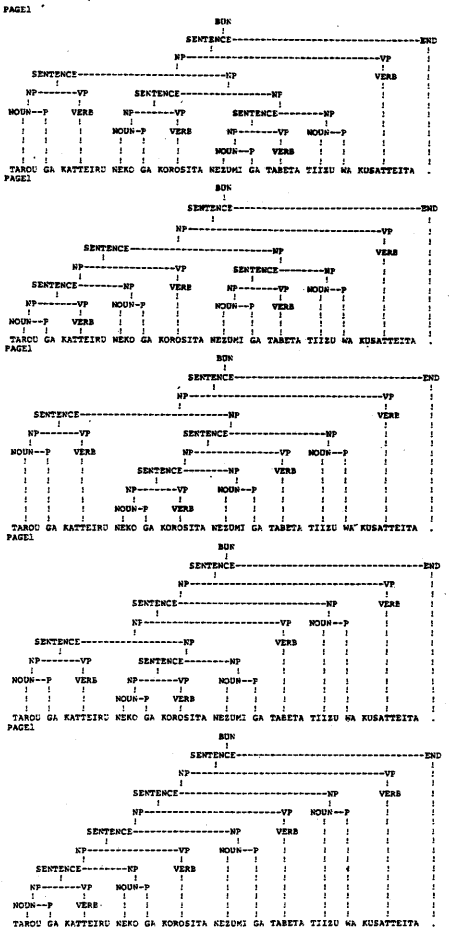


図2 計算機による解析例

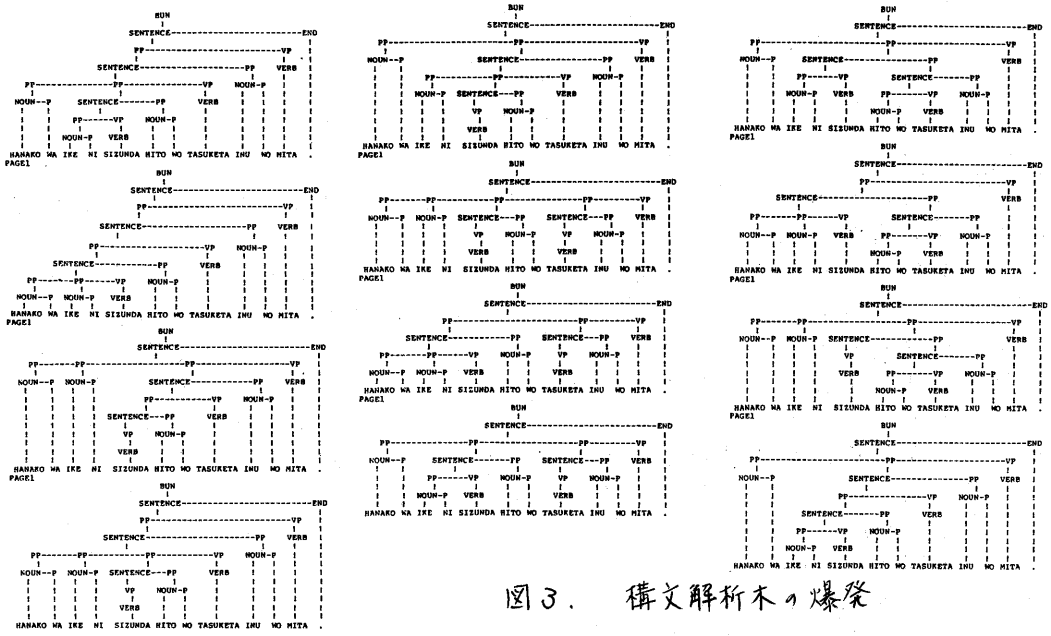


図3. 構文解析木の爆発

文解析木の数の増加は，構文解析に要する時間を増加させるだけでなく意味解析の観点からも，望ましいことではない。

通常このような点が問題とならず，図1に示されるような結果のみが語られる理由は，構文解析木の導出が，計算機によってではなく，人間によって，その文の意味を理解した上で行なわれているからである。従って構文解析を計算機で行なうための文法としては，その視点をはっきりさせた上で，さらに検討することが必要であると思われる。

2. 左方枝分かれ構造の構文解析木

よく知られているように，日本語の埋め込み文は，意味的なあいまいさを含んでいる場合がある。例えば，「花子は川で泳いでいる人を見た。」という文では，「川で見た」，「川で泳いでいる」の2つのあいまいさがある。これは左方枝分かれの構文解析木では，どのように表わされるであろうか。

図4に，前節で用いたものと同様の，左方枝分かれ構造の構文解析木を導出する文法規則を用いた場合の構文解析結果を示す。左方枝分かれ構造を基本とした3種の構文解析木が導出されている。これらの内，現在の例文については，「川で見た」と言う関係を示す(c)と，「川で泳いでいる」と言う関係を示す(a)とが適切な形態

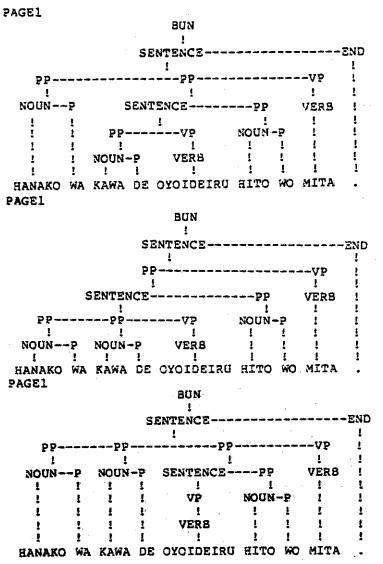


図4. 左方枝分かれ構造の構文解析木

であると思われる。一方(b)が適切な構造となる文としては，「彼女が東京で失な，た物は大きい。」* などが考えられる。

このように左方枝分かれ構造によって導出される構文解析木群は，同じ品詞列を持つ，日本語の埋め込み文が含みうるすべての意味関係を1つ1つの木が示しているわけである。従って，導出された個々の木にはあいまいさがないが，どの木を選ぶかという点にそのあいまいさが移行している。図4の例では高々3種類であり，構文解析木を用いた意味処理の段階でその内の1つを選択してもよいであろうが，図3あるいはさらに複雑な文の場合導出される構文解析木の数が爆発的に増加すれば，収拾がつかなくなる恐れがある。

なお筆者らが現在用いているパーザは拡張LINGOLをn進木の構文解析木が導出できるように改良したものであるから，従来の拡張LINGOLと同様に，advice部を用いた補強文脈自由文法を用いることができる[2]。従って，左方枝分かれ構造においても，不必要な構文解析木の導出を禁止し，構文解析木の種類を減少させることができる。しかしながら，advice部を頻繁に用いることは，構文解析に要する時間を増加させるのみならず，文法の見易さを犠牲にし，さらには文法作成の効率を低下させる。

また，advice部を用いなくとも，非終端記号の種類を増すことにより，構文解析木の数を減らせるのではないか，という議論があるが，この方法は文法規則の数と非終端記号の数を増加させ，構文解析に要する時間を増加させる。また，表層の統語的な役割りによって非終端記号の細分化を行なった場合は次のような問題がある。例えば，後置詞句(Post positional Phrase；PPと略す)を，PP_{GA} や PP_{WO} とすることを考えてみよう。このような場合，「が」や「を」の持つ意味的な機能は，辞書項目の中だけでなく，文法規則中にも記述されることになる。このように，記述を2ヶ所に分離することは，文法作成時，あるいは，辞書項目の追加時に負担がかかる。後置詞としての助詞「が」や「を」が，独自に持つ機能は辞書項目にのみ反映し，文法規則には汎用性があることが望ましい。

さらに高度な構文解析を行なって，句の持つ意味機能を格文法的に細分化しようとすることは，上の理由に加えて，構文解析の初期には，意味情報が不十分であるため，効果があがりにくいと思われる。

3. 右方枝分かれ構造の構文解析木

前節で述べたように，左方枝分かれ構造の構文解析木を導出する文法は，本質的に多数の構文解析木を導出する。では，逆に右方枝分かれ構造の構文解析木を導出するような文法規則

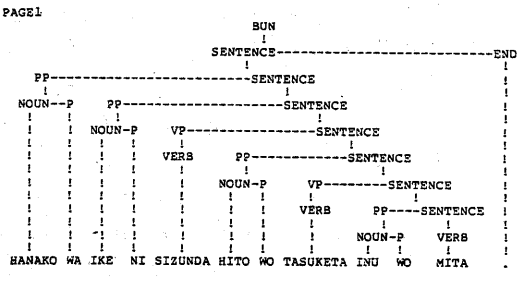


図5. 右方枝分かれ構造の構文解析木(1)

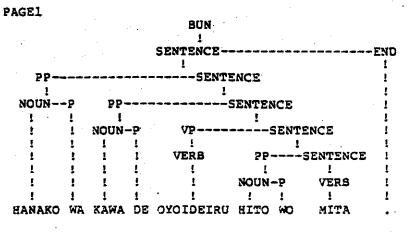


図6. 右方枝分かれ構造の構文解析木(2)

* ((彼女が東京で失なった)物は)大きい)

を用いると，どのような結果が得られるであろうか。

右方枝分かれの構文解析木を導出する文法によって図3，図4と同じ例文を解析した結果を，図5，図6に示す。このように，右方枝分かれ構造の文法規則からは，唯1つの構文解析木しか導出されていない。この構文解析木の構造はどのようなものであるか検討してみよう。

右方枝分かれ構造の構文解析木の特徴は，非終端記号 SENTENCE を根とする部分構文解析木はすべて，1つの日本語文を支配しているということである。例えば，図5に示した例文「花子は池に沈んだ人を助けた犬を見た。」に対する構文解析木を見ると，非終端記号 SENTENCE によって支配される部分はそれぞれ

見た
犬を見た
助けた犬を見た
人を助けた犬を見た
⋮

となっている。これらの部分はすべて，完成した1つの文となっている。これは，この構造の構文解析木が人間の言語直観からしても奇異ではないことを示している。

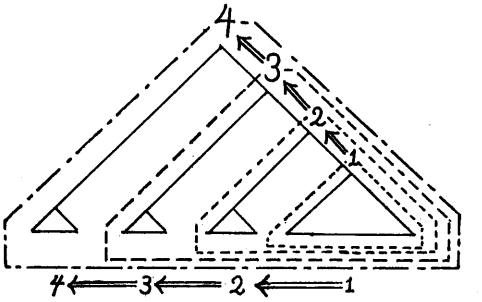


図7．意味処理の広がり

さて，この特徴はこの構造の構文解析木を用いて意味解釈を行なう場合にも有利な事実を示している。すなわち，構文解析木上での意味処理は，図7に示すように，まず小さな部分構文解析木について行ない，その結果を用いて，さらにその上のより大きな部分構文解析木についてこれを繰り返す，という方式が自然に採用できる。これを右方枝分かれ構造の構文解析木について言えば，右から左へと意味処理を進めてゆくことを示している。これは，日本語文からの意味抽出においては，好都合である。日本語は動詞後置の言語であり，格文法などで主張されているように，動詞は名詞などと比べて文の他の構成要素を規定する力が強い。従って，右から左へと処理を進めてゆくと，意味処理の初期の段階で動詞を処理することになり，そこで以後に現われる重要な名詞句をある程度予測できるため，意味抽出を効率的に行なうことができる。

また，構文解析木がそのまま意味解釈を行なうプログラム木となる場合，図8に示すように右方枝分かれ構造の構文解析木においては，日本語で動詞を中心とした意味解析法を採用する場合，この右方枝分かれ構造を右から左にたどる方法により，部分から全体を合成するというフレーゲの原理をはたらかせることができる。これは，関数型言語 LISP の持つ自然な機能を利

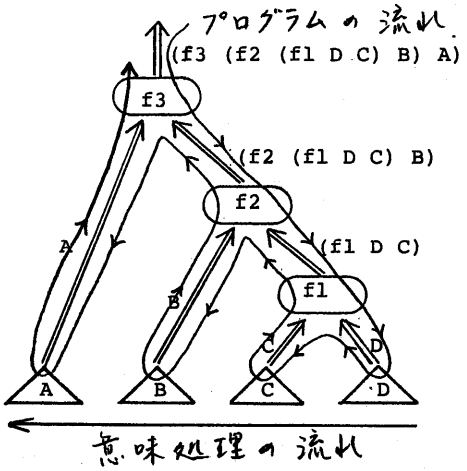


図8．関数性を用いた意味処理
（フレーゲの原理）

用することができる。それにより，後置された動詞の情報や，意味解釈をほどこされた入力文中の右側部分の情報は，自然に上位のnodeに伝播し，さらに大局的な意味処理が行なえる。一方，左方枝分かれ構造の構文解析木において，右から左への意味処理を行なう場合には，文末の動詞の情報をまず最上位のnodeへ持ち込み，その情報を他のnodeが参照しながら意味処理を行なうことになる。言いかえると，最上位におかれたブラックボードのような共通領域に対して構文解析木の各nodeが，情報の書き込み，あるいは参照を行なうことになり，副作用を用いた処理を行なうことになる。これは望ましいことではない。以上の事情を図9に示す。

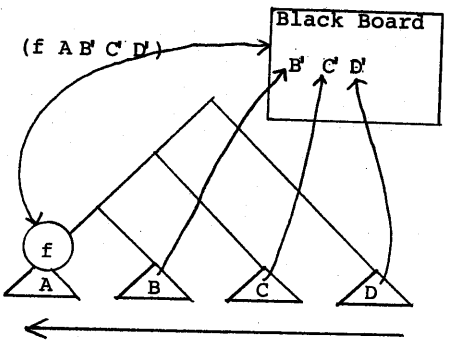


図9．副作用を用いた処理

4．右方枝分かれ構造の構文解析木が内包するあいまいさ

次に，右方枝分かれの構文解析木が，入力された日本語文の持つすべてのあいまいさをimplicitに持っている。言いかえると，左方枝分かれ構造による複数の木の持つ構造を内に含んでいることを示そう。

日本語の処理において，しばしば，係り受けという言葉が用いられるが，構文解析木の上ではどのように定義されるであろうか。筆者らは構文解析木上で，ある構成要素が他の構成要素に係っているための必要条件は，修飾要素が被修飾要素をC－統御*していることであると定義してみた。実際，筆者らが図4の3種の構文解析木から係り受け関係を見出すのは，この考え方による。すなわち，図4の3種の構文解析木の内，(a)(b)では，「見た」は「川で」によってはC－統御できないが，(c)では「見た」は「川で」によってC－統御されている。なお，この条件は必要条件であって十分条件ではないことを注意しておく。

さて，図6に示した右方枝分かれの構文解析木について考えてみよう。ここでは，「花子は」と「川で」の2つの後置詞句は「泳いでいる」と「見た」の両者をC－統御しているのに対し，「人を」は「見た」だけをC－統御している。このことから，「花子は」と「川で」は「泳いでいる」と「見た」の両者に，また，「人を」は「見た」に係りうることがわかる。このように，右方枝分かれ構造の構文解析木（図6）は図4の3種の構文解析木の持つ意味構造をimplicitに内包しているといえる。言いかえると，入力文の品詞並びの持つあいまいさをすべて内包していると言える。

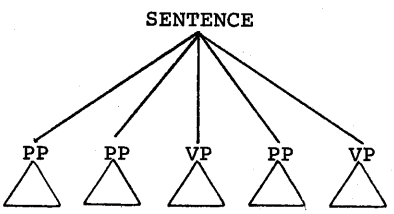
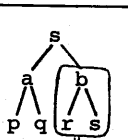


図10．平板な木構造

逆に，この右方枝分かれの構文解析木の持つあいまいさから，この構文解析木が，何ら有意義な構造



aによってC－統御されている。

---

* 節点Aが節点Bを支配せず，かつ，Aを直接支配する節点がBをも支配する時，AはBをC－統御（Constituent Command）するという。

を示していない，すなわち，平板な木構造（図10）と等価ではないかという議論がある。これに対して少なくとも次のことがいえる。右方枝分かれ構造の構文解析木は統語的な語句の並びに対応して幾つかの規則が繰り返し適用され，文の統語構造に応じて作られたものであり，その各規則に対応した幾つかの異なったプログラムにより先に述べたフレーゲの関数原理にそった意味抽出が行なわれるため無構造であるとはいえない。また，平板な木構造の構文解析木での意味処理は，前に述べた左方枝分かれ構造の場合と同様，副作用を用いて行なわねばならない。

また，右方枝分かれ構造の構文解析木が，このようにあいまいさを内包しているので，意味処理の過程での処理に負担がかかる。一方，左方枝分かれ構造の文法規則を用いた場合に得られる構文解析木の数が極端に多い場合はさておき，各構文解析木が相異なった係り受け関係を示すのだから，左方枝分かれ構造による構文解析木を用いた方が有利な場面もあると思われる。しかし，右方枝分かれ構造の構文解析木の持つあいまいさは，入力文中で後置されている動詞の情報などにより，処理途上で減少してゆくため，左方枝分かれ構造の構文解析木群を用いた場合のように手数が，爆発的に増加することはない。

5．非交差の法則の実現

さて，さらに，右方枝分かれ構造のように関数性を用いた処理における係り受け関係の処理について考えてみよう。係り受け関係を決定するための原則の1つとして非交差の法則がある。これは図11に示すように，BD間に係り受け関係がある場合にはAC間には係り受け関係を持てないという規則である。前に述べたC—統御による係り受けの定義ではこの原則を示すことはできない。何故ならばBがDをC—統御しているようにAもCをC—統御しているからである。

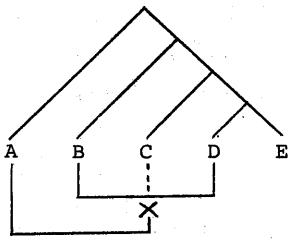


図11．非交差の法則

しかし，実際の意味処理の過程では，構文解析木をプログラムの木として用いているため，部分的な意味処理の結果をプログラムの関数値として上位nodeへ返すことにより，非交差の法則を実現できる。すなわち図12に示すように，S'のnodeに付いたLISPプログラムが，その関数値としてCからの情報を除いた形で返せば，容易にこの法則を実現できる。

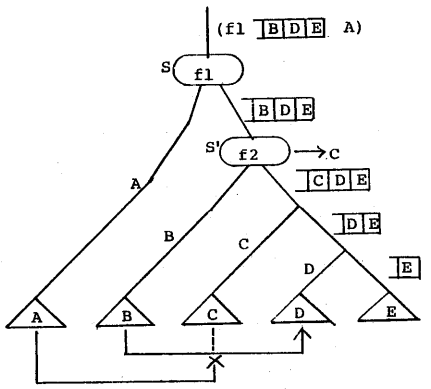


図12．Return Value の制御による非交差の法則の実現

6．構文解析と意味解析の融合法

ここまでの議論で筆者らは暗黙の内に日本語文からの意味抽出の過程を連続する2つの解析過程—構文解析過程と意味解析過程—に分けて行なうものとしてきた。このように日本語文の持つ構文情報と意味情報とを分離して取り扱い，中間表現としての構文解析木によってこの両者をゆるく結合して処理する手法をLCM（Loosely Coupled Model）と呼ぼう。

一方，人間が日本語文を聞いて意味を理

解する場合には，話者の発話につれて意味解釈を進めてゆき，最後の発話と共に文の意味の全容を理解すると思われる。このように発話につれて構文情報と意味情報を密接に関連付けて利用しながら意味抽出を行なう手法をＳＣＭ（Strongly Coupled Model）と呼ぼう。

計算機による意味抽出システムをＳＣＭで実現しようとした場合，入力文に対して構文情報，意味情報を同時に用いながら文法規則を適用して構文解析木を成長させてゆく（図13）。しかし，処理過程の初期にあっては用いることのできる構文情報，意味情報は共に少なく，意味構造を的確に反映した部分構文解析木を作成することには困難が伴う場合がある。実際，埋め込み文では，処理が先に進むまで単語間の係り受けを決定できない場合も多い。次の２文を比較されたい。

（花子が（（川で泳いでいる）少年を）見た。）
（（（花子が（川で泳いでいる））姿を）見た。）

さらに，解析が進まない内は，動詞の情報を用いることが出来ないことも大きく影響する。*

このようにＳＣＭを用いても，ある程度構文解析が進まない内は意味情報を用いた処理は行ないにくく，結局，どこかの時点でBack trackを行なうことになる。**

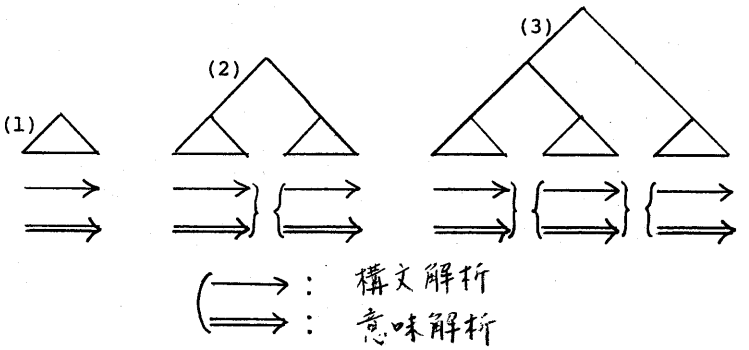


図13．ＳＣＭによる意味抽出過程

ＬＣＭを用いて意味抽出を行なう場合，意味処理過程は，完成した構文解析木に対して行なうため，その方向に制限はない。従って日本語処理の場合，動詞の情報を用いるために，右から左へと，意味処理を進めてゆくことが可能である（図14）。右から左への意味処理においてもBack trackが必要となる場合はある。次の２文を比較されたい。

壁にある絵を見た。
壁がある絵を見た。

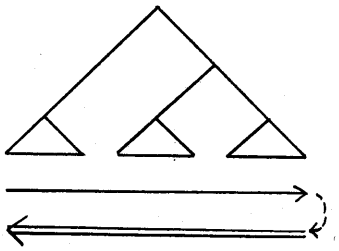


図14．LCMによる意味抽出過程

---

*

もちろん，ＳＣＭにおいて，構文情報・意味情報を同時に用いた解析を，文の末尾から先頭へ向けて行なえば，動詞の情報を最初から用いることもできるが，現在は，発話の順序にあわせて解析を進めてゆくものとして，末尾からの処理の可能性を考慮していない。

**

ＳＣＭで処理を行なう場合にBack trackとは少しく異なった，より人間の思考に近いであろうモデルとして，Marcus，田中の提案しているDiagnosisあるいはDebugの考え方がある[3),4)]。但し，この考え方が計算機処理に適しているかどうかは，さらに検討の必要がある。

しかし，その頻度は左から右への場合と比較すればすくないと思われる。

なお，入力文に対して文法を用いて構文解析木を導出する過程にあっては，右方枝分かれの構文解析規則は，木の自然な成長に沿ったものではない。すなわち，右方枝分かれの構文解析規則では，構文解析が左から右に進むにつれて，小さな部分構文解析木が幾つか作られ，それらが最後の語句の処理と共に一気に結合されて構文解析木を完成させることになる（図15）。このことにより，右方枝分かれの構文解析規則はSCMにはなじまない。一方，左方枝分かれの構文解析規則は，図13からも分かるように部分構文解析木を自然に成長させてゆく形となっている。

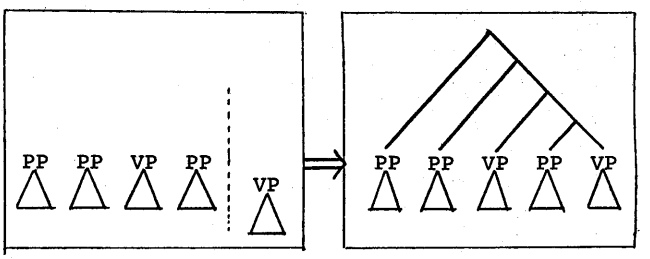


図15．右方枝分かれ構造の構文解析木の導出

7．おわりに

本稿において，筆者らはまず，計算機処理の立場から従来の文法に疑問を投げかけた。次に，特に日本語における埋め込み文について左方枝分かれ構造の構文解析木の問題点を述べた。第3節以下では，その問題点を右方枝分かれ構造が解決する可能性を示した。また，構文解析木の構造と意味抽出の手法の関連にもふれた。このように計算機処理の立場から見ると，右方枝分かれ構造にも幾つかの利点があることが示された。

以上の考察から，現在，筆者らは右方枝分かれ構造を基本構造として日本語の大規模文法の作成を進めている。

参考文献

(1) 久野　暲　；「日本文法研究」　大修館書店　昭．48

(2) 畝見　達夫　；「LINGOLのN進木への拡張」
東京工業大学修士論文　昭．55

(3) Marcus，M. P.；「A Theory of Syntactic Recognition for Natural Language」
The MIT Press　昭．55

(4) 田中　穂積　；「日本語文章解析における二，三の問題」
昭和55年度文部省科学研究費総合研究（A）・第2年次研究報告知識工学の基礎とその応用に関する研究　昭．56

(5) 水谷　静夫　；「機械処理のための日本語文法」
電気学会雑誌　Vol. 93，No. 11，　昭．48