自然言語処理　59－5
（1987. 1. 23）

ＢＵＰ系解析システム上での トップダウンな情報の制御について

奥村学，田村直良，徳永健伸，田中穂積
（東京工業大学　工学部）

純粋にボトムアップなパージングを行なうシステムでは、構文解析木にそって情報を下から上へ送ることができる一方、上から下へ情報を送ることができない。本研究ではまず、ＢＵＰは純粋なボトムアップ法ではなく、ｇｏａｌ節を解析に用いていることで、上から下へ情報を流すことが可能であることを明らかにする。また、この特徴を生かし、上から下へ情報を流す機構ＢＵＰＳを開発した。このＢＵＰＳにより、ボトムアップ解析システム上で、解析中に上下双方向へダイナミックに情報を流すことが可能になることを示す。また、それにより効率よく自然言語解析を行なうことができることを示す。

An Implementation of Top-down Information Passing on BUP System

Manabu OKUMURA, Naoyoshi TAMURA, Takenobu TOKUNAGA and Hozumi TANAKA
(Department of Computer Science, Tokyo Institute of Technology,
2-12-1, Ookayama, Meguro-ku, Tokyo, 152, Japan)

In purely bottom-up parsing system, information can be conveyed upwards naturally, whereas information cannot be conveyed downwards. In this report, first we make it clear that BUP is not purely bottom-up and is capable of passing information downwards by using the 'goal' clause in parsing. Second, we propose the mechanism BUPS passing information downwards, making use of this characteristic of BUP. We also show that BUPS enables information to be conveyed dynamically both upwards and downwards during bottom-up parsing.

1　はじめに

近年、Prologを用いた自然言語解析に関する研究が多く見られるようになってきた。Prologを自然言語解析に用いることの１つの利点は、ＤＣＧ(Definite Clause Grammar)[8]とよばれる文法記述形式で記述した文法を、Prologプログラムに変換し、そのままパーザとして動作させることが可能な点にある。我々は、作成した文法中に左再帰的な文法規則が含まれている場合のことを考慮して、ＤＣＧで記述した文法を変換して、ボトムアップにバージングを行なうPrologプログラムを生成するシステムＢＵＰ[5]を用いている。このＢＵＰはその後、いくつかの改良がなされ[2][3][6]、現在は自然言語解析システムLangLAB[12]の中核になっている。

純粋にボトムアップなバージングを行なうシステムでは、解析木を下から上に向けて生長させるにつれて、情報を下から上へと自然に送ることができる一方、上から下へは情報を送ることができない。そのため、解析途中での構文的，意味的チェックも、情報が下から上がってきてはじめて実行されることになり、上から下に情報を流すことが可能な場合に較べると、その実行のタイミングが遅くなり、不要な計算をすることがある。

また、後述（３章）するように、自然言語解析システムにおいては、下から上，上から下の双方向に情報を流せることが望ましい。文を左から右に読み進みながら理解している人間のモデルを考えるなら、読み終った（理解した）部分の情報を左から右に（解析上は上から下に）流しながら文を解析していき、文を読み終った時点で、理解した結果が解析木の最上部に得られるシステムが、そのシミュレーションとしては一番ふさわしいと思われるからである。

既存のボトムアップ解析システムの中には、下から上への情報を用いて一度バージングを終了し、その後、得られた解析木上をたどる過程で上から下に情報を流すものは存在する[1][10]。しかし、本稿で述べるように、解析途中にダイナミックに情報を双方向に流せるものは存在しなかった。

ＢＵＰは、ボトムアップ解析システムではあるが、詳細に検討すると、純粋なボトムアップ法ではなく、後述する（２章）ｇｏａｌ節をその解析に用いていることから、トップダウンな予測を利用することが可能であり、従って、上から下へ情報を流すことができる。左外置処理のためＢＵＰを拡張したＢＵＰ－ＸＧ[4]では、この特徴を生かし、ｇｏａｌ節を拡張するなどして、左外置処理に必要なスタックを実現している。

本研究では、ＢＵＰ－ＸＧにおけるスタックがストリームとしても用いられることに着目し、同様の手法を用いて、ＢＵＰ系の解析システムでは困難であると思われていた、上から下へ情報を流す機構ＢＵＰＳ(BUP with Stream)を開発した。このＢＵＰＳにより、ボトムアップ解析システム上で、解析途中に上下双方向へダイナミックに情報を流すことが可能になることを示す。

本稿では、まず２章で、ＢＵＰ－ＸＧの概要を述べるとともに、ＢＵＰＳの基本原理を説明する。３章では、簡単な日本語解析システムを例にあげ、ＢＵＰＳがＢＵＰ－ＸＧのスタックを具体的にどのようにストリームとして用いているか、また、ストリームが解析中どのように伝わっていくか述べる。また、４章では、トップダウンな情報の流れを生かした解析の例を示し、ＢＵＰＳが従来のボトムアップ解析システムに較べ、早期に非文法的箇所を検出できるなどの有効性をもつことを示す。

2　ＢＵＰ－ＸＧの概要およびＢＵＰＳの基本原理

ＢＵＰ－ＸＧは、左外置処理をＢＵＰ上で実現したものである。このため、文法の記述形式にも拡張を施したもの（ＸＧＳ）が用いられている。本章では、ＢＵＰ－ＸＧのインプリメント，その文法記述形式ＸＧＳについて簡単に述べるとともに、ＢＵＰＳ（ＢＵＰ－ＸＧ）の基本原理を説明する。なお、ＢＵＰ－ＸＧの詳細は文献[4]を参照して頂きたい。

2.1 ＢＵＰ－ＸＧにおけるスタックの実現法

左外置処理をトップダウンに実行するのに、ＸＧ(extraposition grammar)[9]を用いる方法が考えられている。この方法では、左外置処理実現のため、スタックを用いている。このスタックを実現するため、ＢＵＰ－ＸＧでは、後述するようにＢＵＰの基本動作を利用している。本節では、このスタックの実現法について簡単に説明しておく。

ＢＵＰ－ＸＧでは、スタックの状態を構造体で表わし、それを後続する述語に次々受け渡していくことで、スタックを実現している。例えば、下の文法規則(1)に対して、(2)のように２変数を付加したものを考える。(2)の各文法カテゴリに付加された２変数（下線部）のうち、左の変数は、その文法カテゴリの解析を始める時点でのスタックを表わし、右の変数は、解析が終了した時点でのスタックを表わす。

```
s --> np , vp .                          (1)

s(X0,X2) --> np(X0,X1) , vp(X1,X2) .     (2)
```

これを用いて解析を行なうと（実際の解析は(3)のBUP節で行なわれるから）、

```
np(G,[],I,X0,X1,XR) --> {s(G)},
    goal_x(vp,[],X1,X2),
    s(G,[],I,X0,X2,XR).                  (3)
```

npの解析が成功すると、X0,X1にスタックが返される。npの出力スタックX1が、vpの入力スタックに渡され、vpの解析の結果、X2に新しいスタックが返される。npの入力スタックX0，vpの出力スタックX2をそれぞれsの入出力スタックとして解析を続ける。

ここで注意したいのは、npの出力スタックX1をvpの入力スタックに伝える点である。このvpに対する入力スタックはトップダウンな情報であり、npからｇｏａｌ節のボディで辞書引きされる文法カテゴリまで情報を伝えることができる（図１参照）。これは、上から下に情報を流していることにほかならない。このスタックの流れを実現するメカニズムが、ＢＵＰＳの基本原理となっている。次節では、この基本原理について述べる。

2.2 ＢＵＰＳ（ＢＵＰ－ＸＧ）の基本原理

前節では、ＢＵＰ－ＸＧにおけるスタックの実現法について述べた。ＢＵＰ－ＸＧでは、変数を介して次々とスタックの状態を受け渡し、スタックを実現している。その際、各文法カテゴリに対して入力，出力スタックが存在し、ある文法カテゴリの出力スタックは、文法規則中のその文法カテゴリのすぐ右にある文法カテゴリの入力スタックになっている。これは、文法カテゴリ間のストリームとよべるものである。

以上の考察から、スタックの実現に用いられた２変数をストリームとして利用し、このストリームで情報を左から右に（上から下に）流すことで、より多様な解析システム実現を目指したのがＢＵＰＳである。

なお、ＢＵＰＳでは、文法記述形式としてＸＧＳ（後述）を利用し、同様にトランスレータはＢＵＰ－ＸＧトランスレータ（後述）を用いている。

以下では、このＢＵＰＳの基本原理について述べる。

ＢＵＰＳでは、上に示した(1) → (3)のように、ユーザがＸＧＳで記述した文法は、ＢＵＰ－ＸＧトランスレータによりBUP節等に変換される。そして、それらが図１のｇｏａｌ節とともにボトムアップ構文解析システムとして直接動作する。

解析は、次のゴールを実行することから始まる。

```
?-goal(s,A,[she,smiled],[]) .
```

このゴールの実行により、(4)のボディが実行される。(4)のボディでは、入力文の先頭の単語の辞書引きを行ない（dict述語）、その単語の文法カテゴリをヘッドとするBUP節を選択する。例の場合、先頭の単語"She"の辞書引きの結果、その文法カテゴリであるnpをヘッドとするBUP節(3)が選択される。そして、そのボディの実行において、文法カテゴリvpを次のゴールとして解析を行なう(goal_x(vp,[],X1,X2))。これは、入力文の残りの部分に対して、解析の結果、文法カテゴリvpの部分が存在するはずだという予測をし、その予測（文法カテゴリ名vp）をトップダウンに流していることに相当する。ここで、

```
goal(G,A,X,Z) :-
    dict(C,A1,X,Y),
    Link =.. [C,G],
    call(Link),
    P =.. [C,G,A1,AA,[],[],[],Y,Z],
    call(P),
    A = AA.                                    (4)

goal_x(G,A,X1,X0,X,Z) :-
    dict(C,A1,X,Y),
    Link =.. [C,G],
    call(Link),
    P =.. [C,G,A1,AA,X1,X1,X0,Y,Z],
    call(P),
    A = AA.                                    (5)
```

図１　ＢＵＰＳにおけるｇｏａｌ節

さらにｇｏａｌ節のボディで次の単語を辞書引きし、その文法カテゴリをヘッドとするBUP節を選択することで解析が進むが、このようにＢＵＰは、純粋にボトムアップに解析を行なっているわけではなく、レフトコーナーから解析を始め、そこから得た情報を用いて、次に来るべき文法カテゴリについて予測をし、その予測を（トップダウンに）用いることによって解析を進めているわけである。

また、2.1で述べたように、ストリームが上から下へ流れるのは、図１のgoal_x節(5)において、ヘッドの述語から伝わった入力ストリームX1が、次の単語を辞書引きした結果として呼び出される述語Cの入力ストリームに受け渡されることによる（下線部参照）。すなわち、純粋にストリームは上から下に流れているわけではなく、レフトコーナーから派生したストリームX1がｇｏａｌ節を介して述語Cの呼び出しに伝わることで、次のレフトコーナーに流れるわけである（図２参照）。

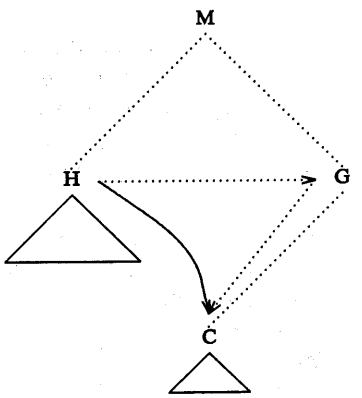


図２　ＢＵＰＳにおけるストリームの流れ

2.3 ＸＧＳおよびＢＵＰ－ＸＧトランスレータ

ＢＵＰ－ＸＧにおける拡張された文法記述形式ＸＧＳで記述された文法は、ＢＵＰＸＧトランスレータによりBUP節に変換される。その際、上で述べたようにストリームを受け渡すための変数をトランスレータが自動的に付加する。従って、通常ユーザはその変数について気にする必要はない。しかし、ユーザが直接変数を操作したいこともある。そのために、ＸＧＳでは、文法規則の中に変数を陽に記述できるようになっている。図３は、ＸＧＳでのその記述例および変換結果である（この例は、文が'and'で等位的に接続された場合には、その接続された２文に対するギャップは同じでなければならないことを記述するためのものである）。

```
s1[X0,X1] ==> s2[X0,X1] , and , s3[X0,X1] .
                ↓変換
s2(G,[],I,X0,X1,XR) --> {s(G)},
      goal(and,[]),
      goal_x(s3,[],X0,X1),
      s1(G,[],I,X0,X1,XR).
```

図３　ＢＵＰ－ＸＧトランスレータによる変換例
（図中の文法カテゴリsに対する添字は、読者に対する便宜でつけたもので、解析上は何の意味ももたない）

例からわかるように、ＸＧＳの記述の[ ]内の２つの変数がXlist用変数としてBUP節にそのまま付加される。

３　ＢＵＰＳを用いた簡単な日本語解析システム

3.1 解析メカニズム

図４に示すＸＧＳで記述した簡単な日本語文法を用いて、「花子は学校へ行った。」という文を解析することにする。

図４の文法を用いると、最終的には図５の解析木が得られるが、その過程でストリームは概略矢印のように流れる。すなわち、後置詞句を解析した意味解釈結果を次々にストリームに追加して文末に渡し、述語（動詞）が現われた直後にそのストリームと動詞の意味を用いて全体の文の意味を決定している。

以下、実際のＢＵＰＳの実行過程を見ることで、具体的にストリームの流れを追うことにする。解析は、

?- goal(文,A,[花子,は,学校,へ,行った],[]).

の呼び出しで開始される。

(1) goal節(4)を用いて、最初の単語「花子」の辞書引きを行ない、その文法カテゴリ「名詞」の規則を選択

(2) 規則(g3)の適用。そのボディの実行において

```
文(S) --> 後置詞句(_) , 文(S) .
文(S)[X0,X1] ==> 述語(Vp) ,
                  {interp((v,t),Vp,X0,S,X1)} .
後置詞句(_)[X0,X1] ==> 名詞(N) , 助詞(P) ,
                  {interp((n,v),P:N,X0,_,X1)}.
                ↓変換
後置詞句(G,_,I,X0,X1,XR) --> {文(G)},
      goal_x(文,S,X1,X2),
      文(G,S,I,X0,X2,XR).          (g1)
述語(G,Vp,I,X0,X0,XR) --> {文(G)},
      {interp((v,t),Vp,X0,S,X1)},
      文(G,S,I,X0,X1,XR).          (g2)
名詞(G,N,I,X0,X0,XR) --> {後置詞句(G)},
      goal(助詞,P),
      {interp((n,v),P:N,X0,_,X1)},
      後置詞句(G,_,I,X0,X1,XR).    (g3)
```

図４　簡単な日本語解析用文法

goal(助詞,P)の呼び出し　……　「は」で成功

(3) 意味解釈プログラムinterpの実行。「花子は」の意味（X:花子）を入力ストリームX0（内容は[]）の先頭に追加してX1に返す

(4) 後置詞句の解析に成功。X1には[X:花子]がバインドされている

(5) 規則(g1)の適用。そのボディの実行において、goal_x(文,S,[X:花子],X2)の呼び出し

(6) 次の単語の辞書引き。以下、(1)～(5)と同様の解析が繰り返され、goal_x(文,S',[へ:学校,X:花子],X2')の呼び出し

(7) 次の単語「行った」の辞書引き。そして、その文法カテゴリ「述語」の呼び出し。この時点で、[へ:学校,X:花子]という情報が、ストリームによりトップダウンに伝わり、利用可能なことに注意してほしい。

(8) 規則(g2)の適用。

(9) 意味解釈プログラムinterpの実行。「行った」の意味(Vp)と入力ストリームX0（[へ:学校,X:花子]）を用いて意味解析。
解析結果
（行った,(行為者:花子,目標:学校,_),1）
をSに返す。X1には[]がバインドされている

(10) トランスレータにより付加された停止条件節を使ってSが次々と上のレベルの「文」に上がり、最終的にトップレベルの「文」の呼び出しのAまで上がってきて、解析終了。解析結果として、
A = (行った,(行為者:花子,目標:学校,_),1)
が得られる。

以上の例から明らかなように、ＢＵＰＳを用いると、「述語」として「行った」が得られた直後で、意味解析

が終了し、全文に対する解析結果が得られる(9)ことに注意してほしい（この点で、ＢＵＰＳが従来のＢＵＰとどのように異なるかについては４章で述べる）。

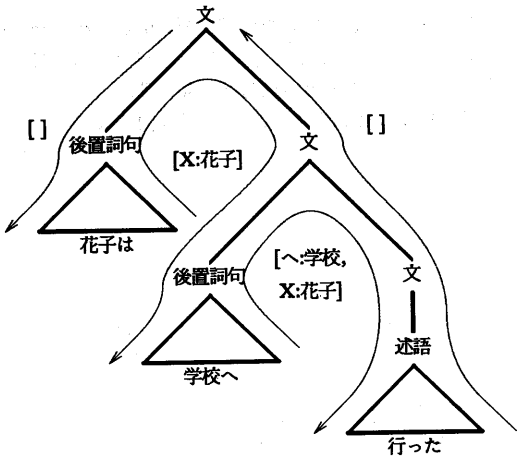


図５　ＢＵＰＳを用いた解析の流れ

3.2 「係り受け非交差の原則」の実現

日本語の解析において重要な役割を果たすのが、文節間の係り受け関係を決定することである。この係り受けに関して一般に日本語には、「係り受け非交差の原則」が成り立つ。

「係り受け非交差の原則」
係りの文節から受けの文節への矢印を用いて係り受け関係を表わすと、その矢印は交差しない。
すなわち、次のようなことはない。

従って、意味解析では、この原則を基に係り受け関係を決定していかなければならない。この原則を意味解析に反映させる具体的方法としては、スタックを用いるのがごく素直であると考えられる。ＢＵＰＳでは、このスタックをストリーム上で、簡潔に実現できる。以下では、例を用いて、そのことを示す。

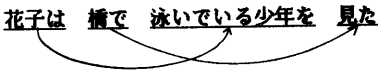
例文として、「花子は橋で泳いでいる少年を見た。」という文を解析することを考える。解析には、図４の文法では不十分で、下のように１つ文法規則を増やした文法を用いる。

```
文(S) --> 後置詞句(_) , 文(S) .                    (g4)
文(S)[X0,X2] ==> 述語(Vp) ,
                 {interp((v,n),Vp,X0,_,X1)} ,
                  文(S)[X1,X2] .                    (g5)
文(S)[X0,X1] ==> 述語(Vp) ,
                 {interp((v,t),Vp,X0,S,X1)} .(g6)
後置詞句(_)[X0,X1] ==> 名詞(N) , 助詞(P) ,
                 {interp((n,v),P:N,X0,_,X1)}.
                                                    (g7)
```

出現する係りの文節（この例の場合には、後置詞句のみ）の意味は、次々ストリームの先頭に付加される(g7)。すなわち、文「花子は橋で泳いでいる少年を見た。」を解析していく過程で、ストリームは、

[]　　（文の先頭）
[X:花子]　　（「花子は」の解析後）
[で:橋,X:花子]　　（「川で」の解析後）

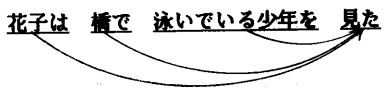
のように変化する。そして、受けの文節として「泳いでいる」が得られた時点で、このストリームを用いて係り受け関係の決定を行なう(g5)。この決定をストリームの先頭から順番に行なっていけば、ストリームはまさにスタックとして動作していることになる（last-inの係りの文節が解析に成功したなら、first-outされる）。この例の場合、「泳いでいる」に対して、その時点でのストリームを用いて、係り受け関係を決定しようとすると、ストリームの先頭の要素である「で:橋」が「泳いでいる」に係りえないので、ストリームの要素はそのままで、後（右）に伝えられる。ここで注意したいのは、ストリームはスタックとして動作しているのであるから、ストリームの先頭要素である「で:橋」が「泳いでいる」に係りえないなら、ストリームのその後の要素に関しては決定を試みないということである。その結果、例えば、次のような、原則に反する係り受け関係は、決して得られない。



この例では、その後、解析が進んだ結果、「見た」が得られる時点でのストリームは、

[を:少年,で:橋,X:花子]

のようになっている。そして、「見た」との係り受け関係の決定を試みた結果、解析は成功して終了する(g6)。その結果、得られるこの例文全体に対する係り受け関係は、次のような妥当なものただ１つとなる。



4　ＢＵＰＳを用いた構文的，意味的チェック

本章では、従来のＢＵＰおよびＢＵＰＳにおける構文的，意味的チェックの実行法を比較することで、その実行のタイミングの違いを明らかにし、ＢＵＰＳの有効性を検討する。なお、構文的チェックの例としては、主語と動詞の呼応のチェックを用いる。

4.1 構文的チェック

（i）従来のＢＵＰの場合

従来のＢＵＰでは、下に示すような文法規則を用いてチェックが行なわれる。

```
s(S_A) --> np(NP_A) , vp(VP_A) ,
          {concord(NP_A,VP_A)} .        (6)
vp(V_A) --> v(V_A) , np(NP_A) .         (7)
```

具体的には、v（動詞）の各単語の辞書に記述されている人称・数に関する情報が、文法規則(7)を用いて述語vpの引数まで伝わり、その結果(6)で、npの人称・数に関する情報との間で呼応のチェックが行なわれる(concord)。これは、呼応のチェックが、(6)のvpの解析終了まで遅延されることを意味する（図６参照）。

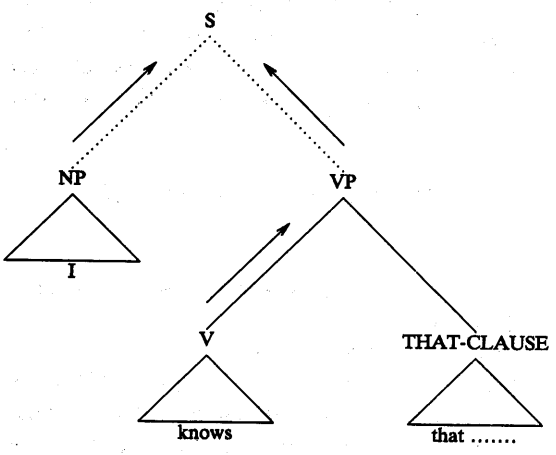


図６　従来のＢＵＰにおける情報の流れ

（ii）ＢＵＰＳの場合

ＢＵＰＳを用いると、次に示すような文法規則でチェックを行なうことができる。

```
s(S_A)[X0,X1] ==> np(NP_A) ,
                  vp(VP_A)[[NP_A|X0],X1].   (8)
vp(V_A)[[NP_A|X0],X0] ==> v(V_A) ,
                          {concord(NP_A,V_A)},
                          np(NP1_A) .       (9)
```

ＢＵＰＳでは、(8)の文法規則で、npの人称・数の情報をストリームに追加してvpに流し、それを用いて(9)の規則中で、vの人称・数の情報との間の呼応のチェックを直ちに行なうことができる(concord)（図７参照）。

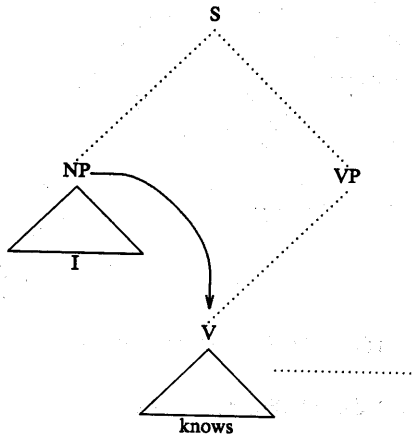


図７　ＢＵＰＳにおける情報の流れ

この例からわかるように、従来のＢＵＰでは、vpの解析が終るまで呼応のチェックが遅延されるのに対し、ＢＵＰＳの場合には、vpの解析中、vが現われた直後に呼応のチェックが行なえる。この違いは、vpにあたる部分が単純な場合には、さほど問題にならないが、vpが複雑で大きな構造をもつ文を解析する場合（動詞が目的語として不定詞句，that節など、文に近いような構造のものをとる場合）には、この呼応のチェックの実行遅延による計算時間の損失は大きくなる。また、同様に、日本語における用言（動詞，形容詞，助動詞等）の間の相互承接の早期チェックにもＢＵＰＳを利用することができる。

4.2 意味的チェック

（i）従来のＢＵＰの場合

従来のＢＵＰでは、下のような文法規則を用いて意味解析を行なう。

```
文(S) --> 後置詞句(Pp) , 文(S1) ,
          {interp(Pp,S1,S)} .           (10)
文(Vp) --> 述語(Vp) .                   (11)
後置詞句(P:N) --> 名詞(N) , 助詞(P) .
```

具体的には、(10)を用いて次々と「後置詞句」の部分

木を生成し、最終的に「述語」が現われた時点で(11)を適用する。この点に関しては、ＢＵＰＳを用いた意味解析（３章参照）と変わらないが、意味解析の実行が(10)のinterpによるため、その実行のタイミングは、「後置詞句」の右側の「文」の意味S1が得られて始めて実行されることになる。従って、後置詞句「花子は」に対して意味解析を実行するのは、その右側の「文」の部分木が完成した後、すなわち、後置詞句「学校へ」と文「行った」から文「学校へ行った」の意味を解析した後ということになる。以上まとめると、従来のＢＵＰの場合には、全体の文に対する意味解析結果が得られるのは、図５における一番上の「文」に対する解析木（全体の文に対する解析木）を形成する時点まで延期される。これは、[7]で有用であると論じられている早期意味解析(early semantic analysis)の考え方に反するものである。

（ii）ＢＵＰＳの場合

ＢＵＰＳを用いた意味解析の詳細については、３章で述べたので、ここでは特に述べないが、まとめると、後置詞句を解析した意味解釈結果を次々にストリームに追加して文末に流し、述語（動詞）が現われた直後にそのストリームと動詞の意味を用いて全体の文の意味を決定している。すなわち、「述語」として「行った」が得られた直後で、意味解析が終了し、全文に対する解析結果が得られる。この時点では、図５のような全体の文に対する解析木が完成しているわけではなく、完成している部分木は、そのうちの、２つの「後置詞句」の部分木および「述語」の部分木だけであることに注意してほしい。

この例から明らかなように、従来のＢＵＰでは、ＢＵＰＳと較べ、意味解析の実行のタイミングが遅くなる。そのため、意味的に異常な箇所の検出が遅くなり、不要な計算を実行してしまう。例えば、３章の例文の「花子は」が「花子を」であるような文「花子を学校へ行った」に対して、ＢＵＰＳの場合には、文末の「行った」の直後の意味解析で、「花子を」が「行った」に係りえないことから、意味的に異常であると判断するのに対し、従来のＢＵＰの場合には、「花子を」が先頭の後置詞句であることから、全体の文に対する解析木を形成する時点で実行される意味解析になって始めて、「行った」に係りえないことに気付くことになる。

以上述べてきたように、ＢＵＰＳを用いれば、構文的，意味的チェックを、従来のＢＵＰに較べ早期に実行できることから、不要な計算を早期に回避することが可能である。

５　おわりに

以上、ＢＵＰＳの基本原理、そのインプリメントなどについて述べ、また、実際の解析例を用いてその解析メカニズムを説明した。また、具体例を用いてその有効性を検討した。

３章で述べた日本語解析システムは、１章で述べた人間の文理解モデルのシミュレーションに、従来のＢＵＰに較べ一歩近づいたといえよう。

また、[11]は、ＢＵＰＳを用いた日本語の並列句解析システムであり、[11]でもＢＵＰＳを用いることの利点が述べられている。

今後の課題としては、３，４章の例からわかるとおり、ＢＵＰＳにおける文法記述はＸＧＳを用いていることもあり、書きづらい。従って、高水準な文法記述言語[10]をＸＧＳの上に設計する必要がある。この問題の解決の方向が、[11]に部分的ながら示されている。

参考文献

[1] 電子技術総合研究所（編）：拡張ＬＩＮＧＯＬ，1978 .

[2] 上脇正，田中穂積，沼崎浩明：ＬａｎｇＬＡＢの構文処理アルゴリズムの高速化，情報処理学会第３３回全国大会論文集，1N-8，1986 .

[3] 今野聡，奥村学，田中穂積：ボトムアップ構文解析システムＢＵＰの高速化，日本ソフトウエア科学会第１回大会論文集，3A-2，1984 .

[4] 今野聡，田中穂積：左外置を考慮したボトムアップ構文解析システム，コンピュータソフトウエア，Vol.3，No.2，1986，pp.19-29 .

[5] Matsumoto,Y.,Tanaka,H.,Hirakawa,H.,Miyoshi,H. and Yasukawa,H.: BUP : A Bottom-Up Parser Embeded in Prolog, New Generation Computing, 1,2, 1983 .

[6] 松本裕治，清野正樹，田中穂積：ＢＵＰの高速化，情報処理学会自然言語処理研究会，39-7，1983 .

[7] Mellish, C.: Computer Interpretation of Natural Language Description, Ellis Horwood Limited, 1985 .

[8] Pereira,F. and Warren,D.: Definite Clause Grammar for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks, Artif. Intell., Vol.13, 1980 .

[9] Pereira,F.: Extraposition Grammar, AJCL, Vol.7, No.4, 1981 .

[10] 田村直良，高倉伸，片山卓也：自然言語処理を目的とした属性文法評価システム，コンピュータソフトウエア，Vol.3，No.3，1986，pp.266-279 .

[11] 田村直良，田中穂積：並列名詞句の構造解析について，情報処理学会自然言語処理研究会，59-2，1987 .
[12] 田中穂積，上脇正，奥村学，沼崎浩明：自然言語処理のためのソフトウエアシステムＬａｎｇＬＡＢ，Proc. of the Logic Programming Conf.'86, 3.1, 1986 .